

EPICS-TwinCAT IOC

2.1

Generated by Doxygen 1.8.18

1 Module Index	1
1.1 Modules	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	11
5.1 File List	11
6 Module Documentation	13
6.1 Device support for TwinCAT/ADS	13
6.1.1 Detailed Description	17
6.1.2 Enumeration Type Documentation	17
6.1.2.1 epics_record_enum	17
6.1.3 Function Documentation	18
6.1.3.1 get_callback_queue_free()	18
6.1.3.2 get_callback_queue_highwatermark()	19
6.1.3.3 get_callback_queue_overflow()	19
6.1.3.4 get_callback_queue_size()	20
6.1.3.5 get_callback_queue_used()	20
6.1.3.6 linkRecord()	21
6.1.3.7 register_devsup()	21
6.1.3.8 set_callback_queue_highwatermark_reset()	22
6.1.4 Variable Documentation	22
6.1.4.1 isCallback	22
6.1.4.2 isPassive	22
6.1.4.3 the_register_devsup	22
6.2 Device driver functions	23
6.2.1 Detailed Description	23
6.3 Info interface classes and functions	24
6.3.1 Detailed Description	24
6.3.2 Typedef Documentation	24
6.3.2.1 info_dbrecord_type	24
6.3.3 Enumeration Type Documentation	24
6.3.3.1 update_enum	24
6.4 TwinCAT tpy file parser	26
6.4.1 Detailed Description	26
6.4.2 Typedef Documentation	26

6.4.2.1 dimension	27
6.4.2.2 dimensions	27
6.4.2.3 enum_map	27
6.4.2.4 enum_pair	27
6.4.2.5 item_list	27
6.4.2.6 symbol_list	28
6.4.2.7 type_multipmap	28
6.4.3 Enumeration Type Documentation	28
6.4.3.1 type_enum	28
6.5 XML tpy file constants	29
6.5.1 Detailed Description	30
6.6 Parser utility functions and classes	31
6.6.1 Detailed Description	32
6.6.2 Typedef Documentation	32
6.6.2.1 property_el	32
6.6.2.2 property_map	32
6.6.2.3 replacement_table	32
6.6.2.4 substitution_map	32
6.6.3 Enumeration Type Documentation	32
6.6.3.1 opc_enum	32
6.6.3.2 process_substitution_enum	33
6.6.3.3 process_tag_enum	33
6.6.3.4 process_type_enum	33
6.7 OPC property constants	35
6.7.1 Detailed Description	36
6.7.2 Variable Documentation	36
6.7.2.1 OPC_NAME_ALIAS	36
6.7.2.2 OPC_PROP_ALIAS	36
6.7.2.3 OPC_PROP_ALMAREAS	36
6.7.2.4 OPC_PROP_ALMCONDITION	37
6.7.2.5 OPC_PROP_ALMCOSV	37
6.7.2.6 OPC_PROP_ALMDB	37
6.7.2.7 OPC_PROP_ALMDEV	37
6.7.2.8 OPC_PROP_ALMFFSV	37
6.7.2.9 OPC_PROP_ALMH	38
6.7.2.10 OPC_PROP_ALMHELP	38
6.7.2.11 OPC_PROP_ALMH	38
6.7.2.12 OPC_PROP_ALMHHSV	38
6.7.2.13 OPC_PROP_ALMHSV	38
6.7.2.14 OPC_PROP_ALML	39
6.7.2.15 OPC_PROP_ALMLIMIT	39
6.7.2.16 OPC_PROP_ALMLL	39

6.7.2.17 OPC_PROP_ALMLLSV	39
6.7.2.18 OPC_PROP_ALMLSV	39
6.7.2.19 OPC_PROP_ALMOSV	40
6.7.2.20 OPC_PROP_ALMPRIMARYAREA	40
6.7.2.21 OPC_PROP_ALMROC	40
6.7.2.22 OPC_PROP_ALMSTAT	40
6.7.2.23 OPC_PROP_ALMUNSV	40
6.7.2.24 OPC_PROP_ALMZRSV	41
6.7.2.25 OPC_PROP_ALMZSV	41
6.7.2.26 OPC_PROP_AVI	41
6.7.2.27 OPC_PROP_BGC	41
6.7.2.28 OPC_PROP_BLINK	41
6.7.2.29 OPC_PROP_BMP	42
6.7.2.30 OPC_PROP_CDT	42
6.7.2.31 OPC_PROP_CLOSE	42
6.7.2.32 OPC_PROP_DESC	42
6.7.2.33 OPC_PROP_DTYP	42
6.7.2.34 OPC_PROP_FFST	43
6.7.2.35 OPC_PROP_FGC	43
6.7.2.36 OPC_PROP_FIELD_BEG	43
6.7.2.37 OPC_PROP_FIELD_END	43
6.7.2.38 OPC_PROP_HIEU	43
6.7.2.39 OPC_PROP_HIRANGE	44
6.7.2.40 OPC_PROP_HTML	44
6.7.2.41 OPC_PROP_INOUT	44
6.7.2.42 OPC_PROP_INPUT	44
6.7.2.43 OPC_PROP_LOEU	44
6.7.2.44 OPC_PROP_LORANGE	45
6.7.2.45 OPC_PROP_OPEN	45
6.7.2.46 OPC_PROP_OUTPUT	45
6.7.2.47 OPC_PROP_PINI	45
6.7.2.48 OPC_PROP_PLNAME	45
6.7.2.49 OPC_PROP_PREC	46
6.7.2.50 OPC_PROP_QUALITY	46
6.7.2.51 OPC_PROP_RECTYPE	46
6.7.2.52 OPC_PROP_RIGHTS	46
6.7.2.53 OPC_PROP_SCANRATE	46
6.7.2.54 OPC_PROP_SERVER	47
6.7.2.55 OPC_PROP SND	47
6.7.2.56 OPC_PROP_TIME	47
6.7.2.57 OPC_PROP_TIMEZONE	47
6.7.2.58 OPC_PROP_TSE	47

6.7.2.59 OPC_PROP_UNIT	48
6.7.2.60 OPC_PROP_VALUE	48
6.7.2.61 OPC_PROP_ZRST	48
6.8 String functions and classes	49
6.8.1 Detailed Description	49
6.8.2 Typedef Documentation	49
6.8.2.1 stringcase	49
6.8.2.2 wstringcase	50
6.8.3 Function Documentation	50
6.8.3.1 compare() [1/2]	50
6.8.3.2 compare() [2/2]	50
6.8.3.3 eq() [1/2]	51
6.8.3.4 eq() [2/2]	51
6.8.3.5 lt() [1/2]	51
6.8.3.6 lt() [2/2]	52
6.8.3.7 ne() [1/2]	52
6.8.3.8 ne() [2/2]	52
6.8.3.9 split_string()	53
6.8.3.10 strncasecmp()	54
6.8.3.11 trim_space() [1/2]	54
6.8.3.12 trim_space() [2/2]	54
6.8.3.13 wcsncasewcmp()	55
6.9 TwinCAT read/write scanning	56
6.9.1 Detailed Description	56
6.10 EPICS utility functions and classes	57
6.10.1 Detailed Description	58
6.10.2 Typedef Documentation	58
6.10.2.1 filename_set	58
6.10.2.2 macro_list	58
6.10.2.3 macro_stack	58
6.10.3 Enumeration Type Documentation	58
6.10.3.1 case_type	58
6.10.3.2 device_support_type	59
6.10.3.3 int_support_type	59
6.10.3.4 io_filestat	59
6.10.3.5 listing_type	60
6.10.3.6 macrofile_type	60
6.10.3.7 string_support_type	60
6.10.3.8 tc_epics_conv	61
6.11 EPICS constants	62
6.11.1 Detailed Description	63
6.11.2 Variable Documentation	63

6.11.2.1 EPICS_DB_ALLOWED	63
6.11.2.2 EPICS_DB_COSV	63
6.11.2.3 EPICS_DB_DESC	63
6.11.2.4 EPICS_DB_DRVH	63
6.11.2.5 EPICS_DB_DRVL	64
6.11.2.6 EPICS_DB_DTYP	64
6.11.2.7 EPICS_DB_EGU	64
6.11.2.8 EPICS_DB_FORBIDDEN	64
6.11.2.9 EPICS_DB_HHSV	64
6.11.2.10 EPICS_DB_HIGH	65
6.11.2.11 EPICS_DB_HIHI	65
6.11.2.12 EPICS_DB_HOPR	65
6.11.2.13 EPICS_DB_HSV	65
6.11.2.14 EPICS_DB_HYST	65
6.11.2.15 EPICS_DB_INP	66
6.11.2.16 EPICS_DB_LLSV	66
6.11.2.17 EPICS_DB_LOLO	66
6.11.2.18 EPICS_DB_LOPR	66
6.11.2.19 EPICS_DB_LOW	66
6.11.2.20 EPICS_DB_LSV	67
6.11.2.21 EPICS_DB_MAJOR	67
6.11.2.22 EPICS_DB_MINOR	67
6.11.2.23 EPICS_DB_NOALARM	67
6.11.2.24 EPICS_DB_NUMVAL	68
6.11.2.25 EPICS_DB_ONAM	68
6.11.2.26 EPICS_DB_OSV	68
6.11.2.27 EPICS_DB_OUT	68
6.11.2.28 EPICS_DB_PINI	68
6.11.2.29 EPICS_DB_PREC	69
6.11.2.30 EPICS_DB_SCAN	69
6.11.2.31 EPICS_DB_SIZV	69
6.11.2.32 EPICS_DB_TSE	69
6.11.2.33 EPICS_DB_UNSV	69
6.11.2.34 EPICS_DB_ZNAM	70
6.11.2.35 EPICS_DB_ZRST	70
6.11.2.36 EPICS_DB_ZRSV	70
6.11.2.37 EPICS_DB_ZRVL	70
6.11.2.38 EPICS_DB_ZSV	71
6.11.2.39 LIGODAQ_DATATYPE_DEFAULT	71
6.11.2.40 LIGODAQ_DATATYPE_FLOAT	71
6.11.2.41 LIGODAQ_DATATYPE_INT32	71
6.11.2.42 LIGODAQ_DATATYPE_NAME	72

6.11.2.43 LIGODAQ_INI_HEADER	72
6.11.2.44 LIGODAQ_UNIT_DEFAULT	72
6.11.2.45 LIGODAQ_UNIT_NAME	72
6.11.2.46 LIGODAQ_UNIT_NONE	73
6.11.2.47 MAX_EPICS_CHANNEL	73
6.11.2.48 MAX_EPICS_DESC	73
6.11.2.49 MAX_EPICS_ENUM	73
6.11.2.50 MAX_EPICS_LONGSTRING	73
6.11.2.51 MAX_EPICS_STRING	74
6.11.2.52 MAX_EPICS_UNIT	74
7 Namespace Documentation	75
7.1 DevTc Namespace Reference	75
7.1.1 Detailed Description	77
7.1.2 Function Documentation	77
7.1.2.1 tc_regex()	77
7.1.2.2 tcAlias()	77
7.1.2.3 tcInfoPrefix()	77
7.1.2.4 tcList()	79
7.1.2.5 tcLoadRecords()	79
7.1.2.6 tcMacro()	79
7.1.2.7 tcPrintVal()	80
7.1.2.8 tcPrintVals()	80
7.1.2.9 tcSetScanRate()	81
7.2 EpicsTpy Namespace Reference	81
7.2.1 Detailed Description	83
7.3 InfoPlc Namespace Reference	83
7.3.1 Detailed Description	84
7.4 ParseTpy Namespace Reference	84
7.4.1 Detailed Description	87
7.4.2 Function Documentation	87
7.4.2.1 compareNamesWoNamespace()	87
7.4.2.2 get_decoration()	87
7.4.2.3 get_pointer()	88
7.5 ParseUtil Namespace Reference	88
7.5.1 Detailed Description	90
7.6 plc Namespace Reference	90
7.6.1 Detailed Description	92
7.6.2 Typedef Documentation	92
7.6.2.1 BasePLCList	92
7.6.2.2 BasePLCPtr	92
7.6.2.3 BaseRecordList	92

7.6.2.4 BaseRecordPtr	93
7.6.2.5 InterfacePtr	93
7.6.2.6 scanner_thread_args	93
7.6.3 Enumeration Type Documentation	93
7.6.3.1 access_rights_enum	93
7.6.3.2 data_type_enum	94
7.6.4 Function Documentation	94
7.6.4.1 reset_and_read() [1/3]	94
7.6.4.2 reset_and_read() [2/3]	95
7.6.4.3 reset_and_read() [3/3]	95
7.6.4.4 ScannerProc()	95
7.6.4.5 scannerThread()	96
7.6.4.6 write_and_test() [1/3]	96
7.6.4.7 write_and_test() [2/3]	96
7.6.4.8 write_and_test() [3/3]	97
7.7 TcComms Namespace Reference	97
7.7.1 Detailed Description	98
7.7.2 Function Documentation	98
7.7.2.1 ADSCallback()	98
7.7.2.2 errorPrintf()	99
7.7.2.3 RouterCall()	99
8 Class Documentation	101
8.1 ParseTpy::ads_routing_info Class Reference	101
8.1.1 Detailed Description	102
8.1.2 Member Function Documentation	102
8.1.2.1 get() [1/2]	102
8.1.2.2 get() [2/2]	103
8.1.2.3 set()	103
8.2 TcComms::AmsRouterNotification Class Reference	104
8.2.1 Detailed Description	104
8.2.2 Friends And Related Function Documentation	104
8.2.2.1 RouterCall	105
8.3 std::atomic< string > Class Reference	105
8.3.1 Detailed Description	106
8.4 std::atomic< wstring > Class Reference	106
8.4.1 Detailed Description	107
8.5 std::atomic_string< stringT > Class Template Reference	107
8.5.1 Detailed Description	108
8.5.2 Member Function Documentation	108
8.5.2.1 operator=()	108
8.6 ParseTpy::base_record Class Reference	109

8.6.1 Detailed Description	110
8.6.2 Constructor & Destructor Documentation	110
8.6.2.1 base_record() [1/4]	110
8.6.2.2 base_record() [2/4]	111
8.6.2.3 base_record() [3/4]	111
8.6.2.4 base_record() [4/4]	111
8.7 plc::BasePLC Class Reference	112
8.7.1 Detailed Description	115
8.7.2 Member Function Documentation	115
8.7.2.1 add() [1/2]	115
8.7.2.2 add() [2/2]	115
8.7.2.3 erase()	116
8.7.2.4 find()	116
8.7.2.5 for_each() [1/2]	117
8.7.2.6 for_each() [2/2]	117
8.7.2.7 get_next()	117
8.7.2.8 get_timestamp_unix()	118
8.7.2.9 plc_data_set_valid()	118
8.7.2.10 printRecord()	118
8.7.2.11 reserve()	119
8.7.2.12 start()	119
8.7.2.13 user_data_set_valid()	119
8.7.3 Member Data Documentation	120
8.7.3.1 records	120
8.8 plc::BaseRecord Class Reference	120
8.8.1 Detailed Description	123
8.8.2 Constructor & Destructor Documentation	123
8.8.2.1 BaseRecord() [1/2]	123
8.8.2.2 BaseRecord() [2/2]	124
8.8.3 Member Function Documentation	124
8.8.3.1 PlcGetValid()	124
8.8.3.2 PlcPush()	124
8.8.3.3 PlcRead() [1/3]	125
8.8.3.4 PlcRead() [2/3]	125
8.8.3.5 PlcRead() [3/3]	126
8.8.3.6 PlcReadBinary()	126
8.8.3.7 PlcSetValid()	126
8.8.3.8 PlcWrite() [1/3]	127
8.8.3.9 PlcWrite() [2/3]	127
8.8.3.10 PlcWrite() [3/3]	128
8.8.3.11 PlcWriteBinary()	128
8.8.3.12 UserGetValid()	129

8.8.3.13 UserPush()	129
8.8.3.14 UserRead() [1/3]	129
8.8.3.15 UserRead() [2/3]	130
8.8.3.16 UserRead() [3/3]	130
8.8.3.17 UserReadBinary()	131
8.8.3.18 UserSetValid()	131
8.8.3.19 UserWrite() [1/3]	132
8.8.3.20 UserWrite() [2/3]	132
8.8.3.21 UserWrite() [3/3]	132
8.8.3.22 UserWriteBinary()	133
8.9 ParseUtil::bit_location Class Reference	134
8.9.1 Detailed Description	135
8.10 std::case_char_traits Struct Reference	135
8.10.1 Detailed Description	136
8.11 std::case_wchar_traits Struct Reference	136
8.11.1 Detailed Description	137
8.12 ParseTpy::compiler_info Class Reference	137
8.12.1 Detailed Description	139
8.13 TcComms::DataPar Struct Reference	139
8.13.1 Detailed Description	139
8.14 plc::DataValue Class Reference	140
8.14.1 Detailed Description	142
8.14.2 Constructor & Destructor Documentation	143
8.14.2.1 DataValue()	143
8.14.3 Member Function Documentation	143
8.14.3.1 GetValid()	143
8.14.3.2 Init()	144
8.14.3.3 PlcGetValid()	144
8.14.3.4 PlcRead() [1/4]	144
8.14.3.5 PlcRead() [2/4]	145
8.14.3.6 PlcRead() [3/4]	145
8.14.3.7 PlcRead() [4/4]	146
8.14.3.8 PlcReadBinary()	146
8.14.3.9 PlcSetValid()	146
8.14.3.10 PlcWrite() [1/4]	147
8.14.3.11 PlcWrite() [2/4]	147
8.14.3.12 PlcWrite() [3/4]	147
8.14.3.13 PlcWrite() [4/4]	148
8.14.3.14 PlcWriteBinary()	148
8.14.3.15 Read() [1/5]	149
8.14.3.16 Read() [2/5]	149
8.14.3.17 Read() [3/5]	149

8.14.3.18 Read() [4/5]	150
8.14.3.19 Read() [5/5]	150
8.14.3.20 ReadBinary()	151
8.14.3.21 SetValid()	151
8.14.3.22 UserGetValid()	151
8.14.3.23 UserRead() [1/4]	152
8.14.3.24 UserRead() [2/4]	152
8.14.3.25 UserRead() [3/4]	153
8.14.3.26 UserRead() [4/4]	153
8.14.3.27 UserReadBinary()	153
8.14.3.28 UserSetValid()	154
8.14.3.29 UserWrite() [1/4]	154
8.14.3.30 UserWrite() [2/4]	154
8.14.3.31 UserWrite() [3/4]	156
8.14.3.32 UserWrite() [4/4]	156
8.14.3.33 UserWriteBinary()	157
8.14.3.34 Write() [1/5]	157
8.14.3.35 Write() [2/5]	158
8.14.3.36 Write() [3/5]	159
8.14.3.37 Write() [4/5]	159
8.14.3.38 Write() [5/5]	160
8.14.3.39 WriteBinary()	160
8.14.4 Member Data Documentation	161
8.14.4.1 mysize	161
8.15 plc::DataValueTraits< T > Struct Template Reference	161
8.15.1 Detailed Description	161
8.16 plc::DataValueTypeDef Struct Reference	162
8.16.1 Detailed Description	164
8.17 DevTc::devTcDefIn< RecType > Struct Template Reference	164
8.17.1 Detailed Description	165
8.18 DevTc::devTcDeflo< RecType > Struct Template Reference	166
8.18.1 Detailed Description	167
8.19 DevTc::devTcDefOut< RecType > Struct Template Reference	168
8.19.1 Detailed Description	169
8.20 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference	169
8.20.1 Detailed Description	170
8.21 EpicsTpy::epics_conversion Class Reference	171
8.21.1 Detailed Description	172
8.21.2 Constructor & Destructor Documentation	172
8.21.2.1 epics_conversion() [1/3]	172
8.21.2.2 epics_conversion() [2/3]	172
8.21.2.3 epics_conversion() [3/3]	173

8.21.3 Member Function Documentation	173
8.21.3.1 getopt()	173
8.21.3.2 to_epics()	174
8.22 EpicsTpy::epics_db_processing Class Reference	175
8.22.1 Detailed Description	176
8.22.2 Constructor & Destructor Documentation	176
8.22.2.1 epics_db_processing()	176
8.22.3 Member Function Documentation	177
8.22.3.1 getopt()	177
8.22.3.2 mygetopt()	177
8.22.3.3 operator()()	178
8.22.3.4 process_field_alarm()	179
8.22.3.5 process_field_numeric() [1/3]	179
8.22.3.6 process_field_numeric() [2/3]	180
8.22.3.7 process_field_numeric() [3/3]	180
8.22.3.8 process_field_string()	180
8.23 EpicsTpy::epics_list_processing Class Reference	181
8.23.1 Detailed Description	182
8.23.2 Constructor & Destructor Documentation	182
8.23.2.1 epics_list_processing() [1/2]	182
8.23.2.2 epics_list_processing() [2/2]	182
8.23.3 Member Function Documentation	183
8.23.3.1 getopt()	183
8.23.3.2 mygetopt()	184
8.23.3.3 operator()()	184
8.24 EpicsTpy::epics_macrofiles_processing Class Reference	185
8.24.1 Detailed Description	186
8.24.2 Constructor & Destructor Documentation	187
8.24.2.1 epics_macrofiles_processing() [1/2]	187
8.24.2.2 epics_macrofiles_processing() [2/2]	187
8.24.3 Member Function Documentation	187
8.24.3.1 getopt()	188
8.24.3.2 mygetopt()	188
8.24.3.3 operator()()	189
8.24.3.4 to_filename()	189
8.25 DevTc::epics_record_traits< RecType > Struct Template Reference	190
8.25.1 Detailed Description	190
8.26 DevTc::epics_tc_db_processing Class Reference	191
8.26.1 Detailed Description	192
8.26.2 Constructor & Destructor Documentation	192
8.26.2.1 epics_tc_db_processing()	192
8.26.3 Member Function Documentation	193

8.26.3.1 getopt()	193
8.26.3.2 operator()()	193
8.26.3.3 process_list()	194
8.26.3.4 process_lists()	194
8.26.3.5 process_macro()	194
8.26.3.6 process_macros()	195
8.27 DevTc::EpicsInterface Class Reference	195
8.27.1 Detailed Description	198
8.28 std::std::hash< std::stringcase > Struct Reference	198
8.28.1 Detailed Description	198
8.28.2 Member Function Documentation	198
8.28.2.1 operator()()	198
8.29 std::std::hash< std::wstringcase > Struct Reference	199
8.29.1 Detailed Description	199
8.29.2 Member Function Documentation	199
8.29.2.1 operator()()	199
8.30 InfoPlc::InfoInterface Class Reference	200
8.30.1 Detailed Description	204
8.30.2 Constructor & Destructor Documentation	205
8.30.2.1 InfoInterface() [1/2]	205
8.30.2.2 InfoInterface() [2/2]	205
8.30.3 Member Function Documentation	205
8.30.3.1 get_infodb()	206
8.30.3.2 get_symbol_name()	206
8.30.3.3 printVal()	206
8.30.4 Member Data Documentation	207
8.30.4.1 dbinfo_list	207
8.31 plc::Interface Class Reference	208
8.31.1 Detailed Description	209
8.31.2 Constructor & Destructor Documentation	209
8.31.2.1 Interface()	209
8.31.3 Member Function Documentation	210
8.31.3.1 printVal()	210
8.32 ParseTpy::item_record Class Reference	210
8.32.1 Detailed Description	211
8.33 EpicsTpy::macro_info Struct Reference	211
8.33.1 Detailed Description	212
8.34 EpicsTpy::macro_record Struct Reference	212
8.34.1 Detailed Description	213
8.35 ParseUtil::memory_location Class Reference	213
8.35.1 Detailed Description	214
8.35.2 Constructor & Destructor Documentation	214

8.35.2.1 memory_location() [1/2]	214
8.35.2.2 memory_location() [2/2]	215
8.35.3 Member Function Documentation	215
8.35.3.1 get()	215
8.35.3.2 set()	215
8.35.3.3 set_section()	216
8.36 EpicsTpy::multi_io_support Class Reference	216
8.36.1 Detailed Description	218
8.36.2 Constructor & Destructor Documentation	218
8.36.2.1 multi_io_support()	218
8.36.3 Member Function Documentation	219
8.36.3.1 getopt()	219
8.37 ParseUtil::opc_list Class Reference	220
8.37.1 Detailed Description	221
8.38 ParseUtil::optarg Class Reference	221
8.38.1 Detailed Description	222
8.38.2 Constructor & Destructor Documentation	222
8.38.2.1 optarg()	222
8.38.3 Member Function Documentation	223
8.38.3.1 parse()	223
8.39 ParseTpy::parserinfo_type Class Reference	224
8.39.1 Detailed Description	226
8.39.2 Member Data Documentation	226
8.39.2.1 name_parse	226
8.40 ParseUtil::parserinfo_type Class Reference	226
8.40.1 Detailed Description	227
8.40.2 Member Data Documentation	227
8.40.2.1 tag	227
8.41 ParseUtil::process_arg Class Reference	228
8.41.1 Detailed Description	229
8.41.2 Constructor & Destructor Documentation	230
8.41.2.1 process_arg()	230
8.41.3 Member Function Documentation	230
8.41.3.1 get_full()	230
8.42 InfoPlc::process_arg_info Class Reference	231
8.42.1 Detailed Description	232
8.42.2 Constructor & Destructor Documentation	232
8.42.2.1 process_arg_info()	232
8.42.3 Member Function Documentation	232
8.42.3.1 get_full()	232
8.43 ParseUtil::process_arg_tc Class Reference	233
8.43.1 Detailed Description	234

8.43.2 Constructor & Destructor Documentation	234
8.43.2.1 process_arg_tc()	234
8.43.3 Member Function Documentation	235
8.43.3.1 get()	235
8.43.3.2 get_full()	235
8.44 ParseTpy::project_record Class Reference	236
8.44.1 Detailed Description	236
8.45 DevTc::register_devsup Class Reference	237
8.45.1 Detailed Description	238
8.46 ParseUtil::replacement_rules Class Reference	238
8.46.1 Detailed Description	239
8.46.2 Constructor & Destructor Documentation	240
8.46.2.1 replacement_rules()	240
8.46.3 Member Function Documentation	240
8.46.3.1 parse_rules()	240
8.47 EpicsTpy::split_io_support Class Reference	241
8.47.1 Detailed Description	243
8.47.2 Constructor & Destructor Documentation	243
8.47.2.1 split_io_support() [1/2]	243
8.47.2.2 split_io_support() [2/2]	244
8.47.3 Member Function Documentation	244
8.47.3.1 getopt()	244
8.47.3.2 increment()	245
8.47.3.3 operator=()	245
8.48 ParseUtil::substitution Class Reference	246
8.48.1 Detailed Description	247
8.48.2 Constructor & Destructor Documentation	247
8.48.2.1 substitution() [1/3]	247
8.48.2.2 substitution() [2/3]	247
8.48.2.3 substitution() [3/3]	247
8.49 ParseUtil::substitution_list Class Reference	248
8.49.1 Detailed Description	249
8.49.2 Member Function Documentation	249
8.49.2.1 query_substitution()	249
8.50 ParseTpy::symbol_record Class Reference	250
8.50.1 Detailed Description	251
8.51 syminfo_processing Class Reference	251
8.51.1 Detailed Description	252
8.52 plc::System Class Reference	252
8.52.1 Detailed Description	253
8.52.2 Member Function Documentation	253
8.52.2.1 add() [1/2]	253

8.52.2.2 add() [2/2]	254
8.52.2.3 for_each() [1/2]	254
8.52.2.4 for_each() [2/2]	254
8.52.2.5 printVal()	255
8.53 ParseUtil::tag_processing Class Reference	255
8.53.1 Detailed Description	256
8.53.2 Constructor & Destructor Documentation	257
8.53.2.1 tag_processing() [1/2]	257
8.53.2.2 tag_processing() [2/2]	257
8.53.3 Member Function Documentation	257
8.53.3.1 getopt()	258
8.54 TcComms::TCatInterface Class Reference	258
8.54.1 Detailed Description	261
8.54.2 Constructor & Destructor Documentation	261
8.54.2.1 TCatInterface()	261
8.54.3 Member Function Documentation	262
8.54.3.1 printVal()	262
8.55 TcComms::TcPLC Class Reference	262
8.55.1 Detailed Description	265
8.55.2 Member Function Documentation	265
8.55.2.1 closePort()	265
8.55.2.2 get_responseBuffer()	266
8.55.2.3 optimizeRequests()	266
8.55.2.4 printRecord()	266
8.55.2.5 set_addr()	267
8.55.2.6 set_ads_state()	267
8.55.2.7 update_scanner()	267
8.55.3 Friends And Related Function Documentation	268
8.55.3.1 ADScallback	268
8.55.4 Member Data Documentation	268
8.55.4.1 cyclesLeft	268
8.56 TcComms::tcProcWrite Class Reference	268
8.56.1 Detailed Description	269
8.56.2 Member Function Documentation	270
8.56.2.1 add()	270
8.56.2.2 read_ptr()	270
8.56.3 Member Data Documentation	271
8.56.3.1 req	271
8.57 DevTc::tcRegisterTolocShell Class Reference	271
8.57.1 Detailed Description	271
8.58 ParseTpy::tpy_file Class Reference	272
8.58.1 Detailed Description	273

8.58.2 Member Function Documentation	273
8.58.2.1 parse_finish()	274
8.58.2.2 process_array()	274
8.58.2.3 process_symbols()	275
8.58.2.4 process_type_tree() [1/3]	276
8.58.2.5 process_type_tree() [2/3]	276
8.58.2.6 process_type_tree() [3/3]	277
8.59 ParseTpy::type_map Class Reference	278
8.59.1 Detailed Description	279
8.60 ParseTpy::type_record Class Reference	279
8.60.1 Detailed Description	281
8.61 ParseUtil::variable_name Class Reference	281
8.61.1 Detailed Description	282
9 File Documentation	283
9.1 atomic_string.h File Reference	283
9.1.1 Detailed Description	283
9.2 devTc.cpp File Reference	284
9.2.1 Detailed Description	285
9.3 devTc.h File Reference	285
9.3.1 Detailed Description	287
9.4 drvTc.cpp File Reference	287
9.4.1 Detailed Description	289
9.5 drvTc.h File Reference	289
9.5.1 Detailed Description	289
9.6 EpicsDbGen.cpp File Reference	290
9.6.1 Detailed Description	290
9.6.2 Function Documentation	290
9.6.2.1 main()	290
9.7 infoPlc.cpp File Reference	291
9.7.1 Detailed Description	291
9.8 infoPlc.h File Reference	291
9.8.1 Detailed Description	292
9.9 infoPlcTemplate.h File Reference	292
9.9.1 Detailed Description	293
9.10 iocMain.cpp File Reference	293
9.10.1 Detailed Description	293
9.10.2 Function Documentation	293
9.10.2.1 main()	294
9.11 ParseTpy.cpp File Reference	294
9.11.1 Detailed Description	295
9.12 ParseTpy.h File Reference	295

9.12.1 Detailed Description	296
9.13 ParseTpyConst.h File Reference	296
9.13.1 Detailed Description	298
9.14 ParseTpyInfo.cpp File Reference	299
9.14.1 Detailed Description	299
9.14.2 Function Documentation	299
9.14.2.1 main()	299
9.15 ParseTpyTemplate.h File Reference	300
9.15.1 Detailed Description	300
9.16 ParseUtil.cpp File Reference	300
9.16.1 Detailed Description	301
9.17 ParseUtil.h File Reference	301
9.17.1 Detailed Description	302
9.18 ParseUtilConst.h File Reference	302
9.18.1 Detailed Description	304
9.19 plcBase.cpp File Reference	304
9.19.1 Detailed Description	305
9.20 plcBase.h File Reference	305
9.20.1 Detailed Description	307
9.21 plcBaseTemplate.h File Reference	307
9.21.1 Detailed Description	307
9.22 stdafx.cpp File Reference	307
9.22.1 Detailed Description	308
9.23 stdafx.h File Reference	308
9.23.1 Detailed Description	308
9.24 stringcase.h File Reference	308
9.24.1 Detailed Description	309
9.25 stringcase_hash.h File Reference	310
9.25.1 Detailed Description	310
9.26 tcComms.cpp File Reference	310
9.26.1 Detailed Description	311
9.27 tcComms.h File Reference	311
9.27.1 Detailed Description	312
9.28 TpyToEpics.cpp File Reference	312
9.28.1 Detailed Description	313
9.29 TpyToEpics.h File Reference	313
9.29.1 Detailed Description	314
9.30 TpyToEpicsConst.h File Reference	315
9.30.1 Detailed Description	316
Index	317

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Device support for TwinCAT/ADS	13
Device driver functions	23
Info interface classes and functions	24
TwinCAT tpy file parser	26
XML tpy file constants	29
Parser utility functions and classes	31
OPC property constants	35
String functions and classes	49
TwinCAT read/write scanning	56
EPICS utility functions and classes	57
EPICS constants	62

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

DevTc	Namespace for TCat device support	75
EpicsTpy	Namespace for tpy-db conversion	81
InfoPlc	Namespace for Info communication	83
ParseTpy	Namespace for parsing	84
ParseUtil	Namespace for parsing utilities	88
plc	Namespace for abstract plc functionality	90
TcComms	Namespace for TCat communication	97

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ParseTpy::ads_routing_info	101
ParseTpy::project_record	236
TcComms::AmsRouterNotification	104
std::atomic_string< stringT >	107
std::atomic_string< string >	107
std::atomic< string >	105
std::atomic_string< wstring >	107
std::atomic< wstring >	106
ParseTpy::base_record	109
ParseTpy::item_record	210
ParseTpy::symbol_record	250
ParseTpy::type_record	279
plc::BasePLC	112
TcComms::TcPLC	262
ParseUtil::bit_location	134
ParseTpy::item_record	210
ParseTpy::type_record	279
char_traits	
std::case_char_traits	135
std::case_wchar_traits	136
ParseTpy::compiler_info	137
ParseTpy::project_record	236
TcComms::DataPar	139
plc::DataValueTraits< T >	161
plc::DataValueTypeDef	162
plc::BaseRecord	120
plc::DataValue	140
DevTc::devTcDeflo< RecType >	166
DevTc::devTcDefIn< RecType >	164
DevTc::devTcDefOut< RecType >	168
DevTc::devTcDefWaveformIn< RecType >	169
DevTc::epics_record_traits< RecType >	190
std::std::hash< std::stringcase >	198

std::std::hash< std::wstringcase >	199
plc::Interface	208
DevTc::EpicsInterface	195
InfoPlc::InfoInterface	200
TcComms::TCatInterface	258
EpicsTpy::macro_info	211
EpicsTpy::macro_record	212
ParseUtil::memory_location	213
ParseTpy::symbol_record	250
EpicsTpy::multi_io_support	216
EpicsTpy::epics_macrofiles_processing	185
ParseUtil::opc_list	220
ParseUtil::optarg	221
ParseTpy::parserinfo_type	224
ParseUtil::parserinfo_type	226
ParseUtil::process_arg	228
InfoPlc::process_arg_info	231
ParseUtil::process_arg_tc	233
DevTc::register_devsup	237
ParseUtil::replacement_rules	238
EpicsTpy::epics_conversion	171
EpicsTpy::epics_db_processing	175
DevTc::epics_tc_db_processing	191
EpicsTpy::epics_list_processing	181
EpicsTpy::epics_macrofiles_processing	185
EpicsTpy::split_io_support	241
EpicsTpy::epics_db_processing	175
EpicsTpy::epics_list_processing	181
ParseUtil::substitution_list	248
EpicsTpy::epics_conversion	171
syminfo_processing	251
plc::System	252
ParseUtil::tag_processing	255
ParseTpy::tpy_file	272
TcComms::tcProcWrite	268
DevTc::tcRegisterTolocShell	271
type_multipmap	
ParseTpy::type_map	278
ParseUtil::variable_name	281
ParseUtil::substitution	246

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ParseTpy::ads_routing_info	ADS routing information	101
TcComms::AmsRouterNotification	AMS Router Notification	104
std::atomic< string >	Atomic<string>	105
std::atomic< wstring >	Atomic<wstring>	106
std::atomic_string< stringT >	Atomic strings	107
ParseTpy::base_record	Base record definition	109
plc::BasePLC	Base PLC	112
plc::BaseRecord	Class for managing a tag/channel	120
ParseUtil::bit_location	Bit location	134
std::case_char_traits	Case insensitive traits	135
std::case_wchar_traits	Case insensitive unicode traits	136
ParseTpy::compiler_info	Compiler information	137
TcComms::DataPar	Memory location struct	139
plc::DataValue	Data value	140
plc::DataValueTraits< T >	Data value traits	161
plc::DataValueTypeDef	Collection of type definitions	162
DevTc::devTcDefIn< RecType >	Device support input record	164
DevTc::devTcDefIo< RecType >	Device support record	166

DevTc::devTcDefOut< RecType >	168
Device support output record	
DevTc::devTcDefWaveformIn< RecType >	169
Device support waveform record	
EpicsTyp::epics_conversion	171
Epics conversion	
EpicsTyp::epics_db_processing	175
Pics database record processing	
EpicsTyp::epics_list_processing	181
List processing	
EpicsTyp::epics_macrofiles_processing	185
Macro file processing	
DevTc::epics_record_traits< RecType >	190
Epics record traits	
DevTc::epics_tc_db_processing	191
EPICS/TCat db processing	
DevTc::EpicsInterface	195
Epics interface class	
std::std::hash< std::stringcase >	198
Hash for case insensitive string	
std::std::hash< std::wstringcase >	199
Hash for case insensitive unicode string	
InfoPlc::InfoInterface	200
Info interface	
plc::Interface	208
Abstract interface	
ParseTyp::item_record	210
Item record	
EpicsTyp::macro_info	211
Macro information	
EpicsTyp::macro_record	212
Macro record	
ParseUtil::memory_location	213
Memory location	
EpicsTyp::multi_io_support	216
Multiple IO support	
ParseUtil::opc_list	220
OPC list	
ParseUtil::optarg	221
Optional arguments	
ParseTyp::parserinfo_type	224
Parser information	
ParseUtil::parserinfo_type	226
Parser information	
ParseUtil::process_arg	228
Arguments for processing	
InfoPlc::process_arg_info	231
Arguments for processing	
ParseUtil::process_arg_tc	233
Arguments for processing	
ParseTyp::project_record	236
Project information	
DevTc::register_devsup	237
Device support registration	
ParseUtil::replacement_rules	238
Replacement rules	
EpicsTyp::split_io_support	241
Split IO support	

ParseUtil::substitution	246
Channel substitution	246
ParseUtil::substitution_list	248
List of channel substitutions	248
ParseTpy::symbol_record	250
Symbol record	250
syminfo_processing	251
Symbol processing	251
plc::System	252
System to keep track of PLCs	252
ParseUtil::tag_processing	255
Tag processing selection	255
TcComms::TCatInterface	258
TCat interface class	258
TcComms::TcPLC	262
TwinCAT PLC	262
TcComms::tcProcWrite	268
TwinCAT process write requests	268
DevTc::tcRegisterTolocShell	271
Register TC commands	271
ParseTpy::tpy_file	272
Tpy file parsing	272
ParseTpy::type_map	278
Type dictionary	278
ParseTpy::type_record	279
Type record information	279
ParseUtil::variable_name	281
Variable name	281

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

atomic_string.h	283
devTc.cpp	284
devTc.h	285
drvTc.cpp	287
drvTc.h	289
EpicsDbGen.cpp	290
infoPlc.cpp	291
infoPlc.h	291
infoPlcTemplate.h	292
iocMain.cpp	293
ParseTpy.cpp	294
ParseTpy.h	295
ParseTpyConst.h	296
ParseTpyInfo.cpp	299
ParseTpyTemplate.h	300
ParseUtil.cpp	300
ParseUtil.h	301
ParseUtilConst.h	302
plcBase.cpp	304
plcBase.h	305
plcBaseTemplate.h	307
stdafx.cpp	307
stdafx.h	308
stringcase.h	308
stringcase_hash.h	310
tcComms.cpp	310
tcComms.h	311
TpyToEpics.cpp	312
TpyToEpics.h	313
TpyToEpicsConst.h	315

Chapter 6

Module Documentation

6.1 Device support for TwinCAT/ADS

Classes

- class `DevTc::EpicsInterface`
Epics interface class.
- struct `DevTc::epics_record_traits< RecType >`
Epics record traits.
- struct `DevTc::devTcDeflo< RecType >`
Device support record.
- struct `DevTc::devTcDefIn< RecType >`
Device support input record.
- struct `DevTc::devTcDefOut< RecType >`
device support output record.
- struct `DevTc::devTcDefWaveformIn< RecType >`
device support waveform record.

Typedefs

- using `DevTc::register_devsup::link_func = bool(&)(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Type descriping the link function.
- using `DevTc::register_devsup::test_pattern = std::pair< std::regex, link_func & >`
pair of pattern and link function
- using `DevTc::register_devsup::test_pattern_list = std::vector< test_pattern >`
list of pattern/link functions
- using `DevTc::epics_record_traits< RecType >::traits_type = struct { double val`
Epics record type.
- using `DevTc::epics_record_traits< RecType >::traits_type_ptr = epics_record_traits< RecType >::value_type *`
Epics record type pointer.
- using `DevTc::epics_record_traits< RecType >::value_type = epicsFloat64`
Value type of (raw) value field.
- using `DevTc::devTcDeflo< RecType >::rec_type = epics_record_traits< RecType >::traits_type`

- Record type: aiRecord, etc.*
- using `DevTc::devTcDeflo< RecType >::rec_type_ptr = epics_record_traits< RecType >::traits_type *`
Pointer to record type.
 - using `DevTc::devTcDeflo< RecType >::report_type = long(*)(int level)`
Report support function.
 - using `DevTc::devTcDeflo< RecType >::init_type = long(*)(int after)`
Initialization function.
 - using `DevTc::devTcDeflo< RecType >::init_record_type = long(*)(rec_type_ptr prec)`
Initialization function.
 - using `DevTc::devTcDeflo< RecType >::get_ioint_info_type = long(*)(int cmd, rec_type_ptr prec, IOSCA* piosl)`
IO/INT support function.
 - using `DevTc::devTcDeflo< RecType >::io_type = long(*)(rec_type_ptr prec)`
IO function like read or write.
 - using `DevTc::devTcDeflo< RecType >::special_linconv_type = long(*)(rec_type_ptr prec, int after)`
Linear conversion support function.
 - using `DevTc::devTcDefIn< RecType >::rec_type = devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
 - using `DevTc::devTcDefIn< RecType >::rec_type_ptr = devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.
 - using `DevTc::devTcDefOut< RecType >::rec_type = devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
 - using `DevTc::devTcDefOut< RecType >::rec_type_ptr = devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.
 - using `DevTc::devTcDefWaveformIn< RecType >::rec_type = devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
 - using `DevTc::devTcDefWaveformIn< RecType >::rec_type_ptr = devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.

Enumerations

- enum `DevTc::epics_record_enum {`
`DevTc::epics_record_enum::aaival = 0, DevTc::epics_record_enum::aaoval, DevTc::epics_record_enum::aival,`
`DevTc::epics_record_enum::aoval,`
`DevTc::epics_record_enum::bival, DevTc::epics_record_enum::boval, DevTc::epics_record_enum::eventval,`
`DevTc::epics_record_enum::histogramval,`
`DevTc::epics_record_enum::longinval, DevTc::epics_record_enum::longoutval, DevTc::epics_record_enum::mbbival,`
`DevTc::epics_record_enum::mabboval,`
`DevTc::epics_record_enum::mbbiDirectval, DevTc::epics_record_enum::mbboDirectval, DevTc::epics_record_enum::stringinval,`
`DevTc::epics_record_enum::stringoutval,`
`DevTc::epics_record_enum::lsival, DevTc::epics_record_enum::lsoval, DevTc::epics_record_enum::waveformval,`
`DevTc::epics_record_enum::airval,`
`DevTc::epics_record_enum::aorval, DevTc::epics_record_enum::birval, DevTc::epics_record_enum::borval,`
`DevTc::epics_record_enum::mbbiDirectval,`
`DevTc::epics_record_enum::mbboDirectval, DevTc::epics_record_enum::mbbirval, DevTc::epics_record_enum::mbborval,`
`DevTc::epics_record_enum::epics_record_enumEnd,`
`DevTc::epics_record_enum::invalidval = -1 }`

Epics record type enum.

Functions

- static void `DevTc::register_devsup::add` (const std::regex &rgx, `link_func` &func) noexcept

Register a pattern/link function.
- static bool `DevTc::register_devsup::linkRecord` (const `std::stringcase` &inpout, dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord) noexcept

linkRecord
- `DevTc::register_devsup::register_devsup` () noexcept

Default constructor (adds linkTcRecord entry)
- `DevTc::register_devsup::register_devsup` (const `register_devsup` &)=`delete`

Disabled copy constructor.
- `DevTc::register_devsup::register_devsup` (`register_devsup` &&)=`delete`

Disabled move constructor.
- `register_devsup` & `DevTc::register_devsup::operator=` (const `register_devsup` &)=`delete`

Disabled assignment operator.
- `register_devsup` & `DevTc::register_devsup::operator=` (`register_devsup` &&)=`delete`

Disabled move assignment operator.
- `DevTc::EpicsInterface::EpicsInterface` (plc::BaseRecord &dval) noexcept

Constructor.
- void `DevTc::EpicsInterface::set_isPassive` (bool passive) noexcept

Set isPassive.
- bool `DevTc::EpicsInterface::get_isCallback` () const noexcept

Get isCallback.
- void `DevTc::EpicsInterface::set_isCallback` (bool isCb) noexcept

Set isCallback.
- void `DevTc::EpicsInterface::set_pEpicsRecord` (dbCommon *pEpRecord) noexcept

Set pEpicsRecord.
- bool `DevTc::EpicsInterface::get_callbackRequestPending` () const noexcept

Get callbackRequestPending.
- const epicsCallback & `DevTc::EpicsInterface::callback` () const noexcept

Get pointer to callback structure.
- epicsCallback & `DevTc::EpicsInterface::callback` () noexcept

Get pointer to callback structure.
- const IOSCANPVT & `DevTc::EpicsInterface::ioscan` () const noexcept

Get reference to io scan list pointer.
- IOSCANPVT & `DevTc::EpicsInterface::ioscan` () noexcept

Get reference to io scan list pointer.
- IOSCANPVT `DevTc::EpicsInterface::get_ioscan` () const noexcept

Get pointer to io scan list.
- void `DevTc::EpicsInterface::set_ioscan` (const IOSCANPVT ioscan) noexcept

Set pointer to io scan list.
- bool `DevTc::EpicsInterface::push` () noexcept override

Makes a call to the EPICS dbProcess function.
- bool `DevTc::EpicsInterface::pull` () noexcept override

Does nothing.
- static int `DevTc::EpicsInterface::get_callback_queue_size` (int pri) noexcept
- static int `DevTc::EpicsInterface::get_callback_queue_used` (int pri) noexcept
- static int `DevTc::EpicsInterface::get_callback_queue_free` (int pri) noexcept
- static int `DevTc::EpicsInterface::get_callback_queue_highwatermark` (int pri) noexcept
- static int `DevTc::EpicsInterface::get_callback_queue_overflow` (int pri) noexcept
- static int `DevTc::EpicsInterface::set_callback_queue_highwatermark_reset` () noexcept

- void `DevTc::EpicsInterface::ioscan_reset` (int bitnum) noexcept

Reset ioscan use flag.
- static const char *const `DevTc::epics_record_traits< RecType >::name` ()

Name of the record.
- static `traits_type_ptr DevTc::epics_record_traits< RecType >::val` (`traits_type *prec`)

Returns the (raw) value of a record.
- static bool `DevTc::epics_record_traits< RecType >::read` (`traits_type_ptr epicsrec, plc::BaseRecord *baserec`)

Performs the read access on prec.
- static bool `DevTc::epics_record_traits< RecType >::write` (`plc::BaseRecord *baserec, traits_type_ptr epicsrec`)

Performs the write access on prec.
- `DevTc::devTcDeflo< RecType >::devTcDeflo` (`init_record_type ioinit, io_type io`) noexcept

Constructor for IO record.
- static long `DevTc::devTcDeflo< RecType >::get_ioint_info` (int cmd, `rec_type_ptr prec, IOSCANPVT *ppvt`) noexcept

IO/INT info callback.
- `DevTc::devTcDefIn< RecType >::devTcDefIn` () noexcept

Constructor.
- static long `DevTc::devTcDefIn< RecType >::init_read_record` (`rec_type_ptr prec`) noexcept

init callback for read records
- static long `DevTc::devTcDefIn< RecType >::read` (`rec_type_ptr precord`) noexcept

read callback
- `DevTc::devTcDefOut< RecType >::devTcDefOut` () noexcept

Constructor.
- static long `DevTc::devTcDefOut< RecType >::init_write_record` (`rec_type_ptr prec`) noexcept

init callback for write records
- static long `DevTc::devTcDefOut< RecType >::write` (`rec_type_ptr precord`) noexcept

write callback
- `DevTc::devTcDefWaveformIn< RecType >::devTcDefWaveformIn` () noexcept

Constructor.
- static long `DevTc::devTcDefWaveformIn< RecType >::init_read_waveform_record` (`rec_type_ptr prec`) noexcept

init callback for read records
- static long `DevTc::devTcDefWaveformIn< RecType >::read_waveform` (`rec_type_ptr precord`) noexcept

read callback
- void `DevTc::complete_io_scan` (`EpicsInterface *epics, IOSCANPVT ioscan, int prio`) noexcept

Variables

- `test_pattern_list` `DevTc::register_devsup::tp_list`

list of pattern and links
- static `register_devsup` `DevTc::register_devsup::the_register_devsup`

the one global instance of the register class
- bool `DevTc::EpicsInterface::isPassive` = false
- bool `DevTc::EpicsInterface::isCallback` = false
- `dbCommon * DevTc::EpicsInterface::pEpicsRecord` = nullptr

Pointer to the EPICS record.
- std::mutex `DevTc::EpicsInterface::ioscanmux`

IOSCAN mutex.
- IOSCANPVT `DevTc::EpicsInterface::ioscanpvt` = nullptr

- std::atomic< unsigned int > **DevTc::EpicsInterface::ioscan_inuse** = 0
 - Scan in progress (bit encoded value from priorities)*
- epicsCallback **DevTc::EpicsInterface::callbackval** = {}
 - Callback structure.*
- static const int **DevTc::epics_record_traits< RecType >::value_conversion** = 0
 - return value for read_io functions 0=default, 2=don't convert*
- static const bool **DevTc::epics_record_traits< RecType >::input_record** = true
 - Indicates if this is an input record.*
- static const bool **DevTc::epics_record_traits< RecType >::raw_record** = false
 - Indicates if this is a raw record.*
- long **DevTc::devTcDeflo< RecType >::number** = 6
 - Number of support functions.*
- **report_type DevTc::devTcDeflo< RecType >::report_fn** = nullptr
 - Report support function.*
- **init_type DevTc::devTcDeflo< RecType >::init_fn** = nullptr
 - Init support function.*
- **init_record_type DevTc::devTcDeflo< RecType >::init_record_fn** = nullptr
 - Record init support function.*
- **get_ioint_info_type DevTc::devTcDeflo< RecType >::get_ioint_info_fn** = **get_ioint_info**
 - IO/INT support function.*
- **io_type DevTc::devTcDeflo< RecType >::io_fn** = nullptr
 - Read/write support function.*
- **special_linconv_type DevTc::devTcDeflo< RecType >::special_linconv_fn** = nullptr
 - Linear conversion support function.*

Friends

- void **DevTc::EpicsInterface::complete_io_scan** (**EpicsInterface** *, IOSCANPVT, int) noexcept

6.1.1 Detailed Description

6.1.2 Enumeration Type Documentation

6.1.2.1 epics_record_enum

```
enum DevTc::epics_record_enum [strong]
```

Epics record type enum.

This record type enums are used as index the epics traits class

Enumerator

aaival	double input array
aaoval	double output array
aival	double input

Enumerator

aoval	double output
bival	binary input
boval	binary output
eventval	event
histogramval	histogram
longinval	integer input
longoutval	integer output
mbbival	enum input
mbboval	enum output
mbbiDirectval	enum input direct
mbboDirectval	enum output direct
stringinval	string input
stringoutval	string output
lsival	long string input
lsoval	long string output
waveformval	waveform
airval	raw double input
aorval	raw double output
birval	raw binary input
borval	raw binary output
mbbiDirectrvl	raw enum input
mbboDirectrvl	raw enum output
mbbirval	raw enum input direct
mbborval	raw enum output direct
epics_record_enumEnd	End of enum (sentinel value)
invalidval	invalid

Definition at line 232 of file devTc.h.

6.1.3 Function Documentation

6.1.3.1 get_callback_queue_free()

```
int DevTc::EpicsInterface::get_callback_queue_free (
    int pri ) [static], [noexcept]
```

Get the free entries in the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

in src\ioc\db\callback.c (not needed in EPICS 7)

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

free entries in the callback ring buffer

Definition at line 355 of file devTc.cpp.

6.1.3.2 get_callback_queue_highwatermark()

```
int DevTc::EpicsInterface::get_callback_queue_highwatermark (
    int pri ) [static], [noexcept]
```

Get the high watermark in the callback ring buffer

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

high watermark in the callback ring buffer

Definition at line 375 of file devTc.cpp.

6.1.3.3 get_callback_queue_overflow()

```
int DevTc::EpicsInterface::get_callback_queue_overflow (
    int pri ) [static], [noexcept]
```

Get the number of overflows in the callback ring buffer

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

number of overflows in the callback ring buffer

Definition at line 395 of file devTc.cpp.

6.1.3.4 get_callback_queue_size()

```
int DevTc::EpicsInterface::get_callback_queue_size (
    int pri ) [static], [noexcept]
```

Get the size of the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

in src\ioc\db\callback.c (not needed in EPICS 7)

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

size of the callback ring buffer

Definition at line 313 of file devTc.cpp.

6.1.3.5 get_callback_queue_used()

```
int DevTc::EpicsInterface::get_callback_queue_used (
    int pri ) [static], [noexcept]
```

Get the used entries in the callback ring buffer For this function to return a valid value the EPICS distribution needs to be patched. Add the following lines:

```
epicsShareFunc epicsRingPointerId tcat_callbackQueue (int Priority)
{
    return (Priority >= 0) && (Priority < NUM_CALLBACK_PRIORITIES) ?
        callbackQueue[Priority].queue : NULL;
}
```

after the declaration of

```
static cbQueueSet callbackQueue[NUM_CALLBACK_PRIORITIES];
```

in src\ioc\db\callback.c (not needed in EPICS 7)

Parameters

<i>pri</i>	Priority of ring buffer
------------	-------------------------

Returns

used entries in the callback ring buffer

Definition at line 333 of file devTc.cpp.

6.1.3.6 linkRecord()

```
bool DevTc::register_devsup::linkRecord (
    const std::string& inout,
    dbCommon * pEpicsRecord,
    plc::BaseRecordPtr & pRecord ) [static], [noexcept]
```

linkRecord

Go through list and call first link function which matches the pattern Used to link epics records with internal records.

Parameters

<i>inout</i>	Value of INP/OUT field
<i>pEpicsRecord</i>	Pointer to EPICS record
<i>pRecord</i>	Pointer to a base record (return)

Returns

true if one match was found and successfully linked

register_devsup::linkRecord

Definition at line 110 of file devTc.cpp.

References plc::System::find(), and plc::System::get().

Here is the call graph for this function:

6.1.3.7 register_devsup()

```
DevTc::register_devsup::register_devsup ( ) [protected], [noexcept]
```

Default constructor (adds linkTcRecord entry)

register_devsup::register_devsup

Definition at line 102 of file devTc.cpp.

References DevTc::tc_regex().

Here is the call graph for this function:

6.1.3.8 `set_callback_queue_highwatermark_reset()`

```
int DevTc::EpicsInterface::set_callback_queue_highwatermark_reset ( ) [static], [noexcept]
```

Reset the overflow count in the callback ring buffer

Returns

number of overflows in the callback ring buffer

Definition at line 415 of file devTc.cpp.

6.1.4 Variable Documentation

6.1.4.1 `isCallback`

```
bool DevTc::EpicsInterface::isCallback = false [protected]
```

Bool indicating whether callback is needed to call dbProcess true : SCAN = I/O Intr or the record is an out record

Definition at line 215 of file devTc.h.

6.1.4.2 `isPassive`

```
bool DevTc::EpicsInterface::isPassive = false [protected]
```

Bool indicating passive scan true : EPICS record SCAN field is set to PASSIVE

Definition at line 212 of file devTc.h.

6.1.4.3 `the_register_devsup`

```
register_devsup DevTc::register_devsup::the_register_devsup [static], [protected]
```

the one global instance of the register class

[register_devsup::the_register_devsup](#)

Definition at line 81 of file devTc.h.

Referenced by DevTc::register_devsup::add().

6.2 Device driver functions

Namespaces

- [DevTc](#)

Namespace for TCat device support.

6.2.1 Detailed Description

6.3 Info interface classes and functions

Classes

- class `InfoPlc::InfoInterface`
Info interface.
- class `InfoPlc::process_arg_info`
Arguments for processing.

Typedefs

- using `InfoPlc::info_update_method` = `bool(InfoInterface::*)()`
Pointer to info update method.
- using `InfoPlc::info_dbrecord_type` = `std::tuple< ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method >`
- using `InfoPlc::info_dbrecord_list` = `std::vector< info_dbrecord_type >`
List type of db info tuples.

Enumerations

- enum `InfoPlc::update_enum` { `InfoPlc::update_enum::forever`, `InfoPlc::update_enum::once`, `InfoPlc::update_enum::done` }
- Update frequency type.*

6.3.1 Detailed Description

6.3.2 Typedef Documentation

6.3.2.1 `info_dbrecord_type`

```
using InfoPlc::info_dbrecord_type = typedef std::tuple<ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method>
```

db info tuple with variable name, process type enum, opc list, TwinCAT type string, readonly, update enum, and update method

Definition at line 42 of file infoPlc.h.

6.3.3 Enumeration Type Documentation

6.3.3.1 `update_enum`

```
enum InfoPlc::update_enum [strong]
```

Update frequency type.

Enumerator

forever	Repeat foreveer.
once	once
done	done

Definition at line 27 of file infoPlc.h.

6.4 TwinCAT tpy file parser

Classes

- class [ParseTpy::ads_routing_info](#)
ADS routing information.
- class [ParseTpy::compiler_info](#)
Compiler information.
- class [ParseTpy::project_record](#)
Project information.
- class [ParseTpy::base_record](#)
Base record definition.
- class [ParseTpy::item_record](#)
item record
- class [ParseTpy::type_record](#)
Type record information.
- class [ParseTpy::type_map](#)
Type dictionary.
- class [ParseTpy::symbol_record](#)
Symbol record.
- class [ParseTpy::tpy_file](#)
Tpy file parsing.

Typedefs

- using [ParseTpy::dimension](#) = std::pair< int, int >
- using [ParseTpy::dimensions](#) = std::list< [dimension](#) >
- using [ParseTpy::enum_map](#) = std::map< int, [std::stringcase](#) >
- using [ParseTpy::enum_pair](#) = std::pair< int, [std::stringcase](#) >
- using [ParseTpy::item_list](#) = std::list< [item_record](#) >
- using [ParseTpy::type_multimap](#) = std::multimap< unsigned int, [type_record](#) >
- using [ParseTpy::symbol_list](#) = std::list< [symbol_record](#) >

Enumerations

- enum [ParseTpy::type_enum](#) {
 [ParseTpy::type_enum::unknown](#), [ParseTpy::type_enum::simple](#), [ParseTpy::type_enum::arraytype](#), [ParseTpy::type_enum::enumtype](#),
[ParseTpy::type_enum::structtype](#), [ParseTpy::type_enum::functionblock](#) }
- Type enum.*

6.4.1 Detailed Description

6.4.2 Typedef Documentation

6.4.2.1 dimension

```
using ParseTpy::dimension = typedef std::pair<int, int>
```

This class stores a lbound, elements pair.

Definition at line 219 of file ParseTpy.h.

6.4.2.2 dimensions

```
using ParseTpy::dimensions = typedef std::list<dimension>
```

This list stores lbound, elements pairs.

Definition at line 223 of file ParseTpy.h.

6.4.2.3 enum_map

```
using ParseTpy::enum_map = typedef std::map<int, std::stringcase>
```

This map stores a list of enum values.

Definition at line 227 of file ParseTpy.h.

6.4.2.4 enum_pair

```
using ParseTpy::enum_pair = typedef std::pair<int, std::stringcase>
```

This type stores an enum pair.

Definition at line 231 of file ParseTpy.h.

6.4.2.5 item_list

```
using ParseTpy::item_list = typedef std::list<item_record>
```

This class stores a list of subitems.

Definition at line 245 of file ParseTpy.h.

6.4.2.6 symbol_list

```
using ParseTpy::symbol_list = typedef std::list<symbol_record>
```

This is a list of symbol records

Definition at line 353 of file ParseTpy.h.

6.4.2.7 type_multimap

```
using ParseTpy::type_multimap = typedef std::multimap<unsigned int, type_record>
```

This is a multimap to store type records

Definition at line 314 of file ParseTpy.h.

6.4.3 Enumeration Type Documentation

6.4.3.1 type_enum

```
enum ParseTpy::type_enum [strong]
```

Type enum.

This structure describes a type record

Enumerator

unknown	Unknown type.
simple	Simple type.
arraytype	Array type.
enumtype	Enumerated type.
structtype	Structure type.
functionblock	Function block.

Definition at line 251 of file ParseTpy.h.

6.5 XML tpy file constants

Variables

- const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"
PLC project info.
- const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"
Project info.
- const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"
Routing info.
- const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"
Compiler info.
- const char *const ParseTpy::xmlAdsInfo = "AdsInfo"
ADS info.
- const char *const ParseTpy::xmlDataTypes = "DataTypes"
Data types.
- const char *const ParseTpy::xmlDataType = "DataType"
Data type.
- const char *const ParseTpy::xmlSymbols = "Symbols"
Symbols.
- const char *const ParseTpy::xmlSymbol = "Symbol"
Symbol.
- const char *const ParseTpy::xmlProperties = "Properties"
Properties.
- const char *const ParseTpy::xmlProperty = "Property"
Property.
- const char *const ParseTpy::xmlCompilerVersion = "CompilerVersion"
Compiler version.
- const char *const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"
TwinCAT version.
- const char *const ParseTpy::xmlCpuFamily = "CpuFamily"
CPU family.
- const char *const ParseTpy::xmlNetId = "NetId"
Net ID.
- const char *const ParseTpy::xmlPort = "Port"
Port.
- const char *const ParseTpy::xmlTargetName = "TargetName"
Target name.
- const char *const ParseTpy::xmlName = "Name"
Name.
- const char *const ParseTpy::xmlType = "Type"
Type.
- const char *const ParseTpy::xmlAttrDecoration = "Decoration"
Decoration.
- const char *const ParseTpy::xmlAttrPointer = "Pointer"
Pointer.
- const char *const ParseTpy::xmlIGroup = "IGroup"
I Group.
- const char *const ParseTpy::xmlIOffset = "IOffset"
I Offset.

- const char *const ParseTpy::xmlBitSize = "BitSize"
Bit size.
- const char *const ParseTpy::xmlBitOffs = "BitOffs"
Bit Offset.
- const char *const ParseTpy::xmlArrayInfo = "ArrayInfo"
Array info.
- const char *const ParseTpy::xmlArrayLBound = "LBound"
Lower bound.
- const char *const ParseTpy::xmlArrayElements = "Elements"
Elements.
- const char *const ParseTpy::xmlSubItem = "SubItem"
Sub item.
- const char *const ParseTpy::xmlFbInfo = "FbInfo"
Fb info.
- const char *const ParseTpy::xmlEnumInfo = "EnumInfo"
Enum info.
- const char *const ParseTpy::xmlEnumText = "Text"
Text.
- const char *const ParseTpy::xmlEnumEnum = "Enum"
Enum.
- const char *const ParseTpy::xmlEnumComment = "Comment"
Comment.
- const char *const ParseTpy::xmlValue = "Value"
Value.
- const char *const ParseTpy::xmlDesc = "Desc"
Description.
- const char *const ParseTpy::opcExport = "opc"
OPC.
- const char *const ParseTpy::opcProp = "opc_prop"
OPC property.
- const char *const ParseTpy::opcBracket = "["
OPC bracket.
- const char *const ParseTpy::xmlSubstitution = "TcSubstitution"
Substitution.
- const char *const ParseTpy::xmlAlias = "Alias"
Substitution EPICS channel name.

6.5.1 Detailed Description

6.6 Parser utility functions and classes

Classes

- class `ParseUtil::replacement_rules`
Replacement rules.
- class `ParseUtil::optarg`
Optional arguments.
- class `ParseUtil::opc_list`
OPC list.
- class `ParseUtil::variable_name`
Variable name.
- class `ParseUtil::bit_location`
Bit location.
- class `ParseUtil::memory_location`
Memory location.
- class `ParseUtil::process_arg`
Arguments for processing.
- class `ParseUtil::process_arg_tc`
Arguments for processing.
- class `ParseUtil::tag_processing`
Tag processing selection.
- class `ParseUtil::substitution`
Channel substitution.
- class `ParseUtil::substitution_list`
List of channel substitutions.

Typedefs

- using `ParseUtil::replacement_table` = `std::map< std::stringcase, std::stringcase >`
- using `ParseUtil::property_map` = `std::map< int, std::stringcase >`
- using `ParseUtil::property_el` = `std::pair< int, std::stringcase >`
- using `ParseUtil::substitution_map` = `std::map< std::stringcase, substitution >`
Map of channel substitution.

Enumerations

- enum `ParseUtil::opc_enum` { `ParseUtil::opc_enum::no_change`, `ParseUtil::opc_enum::publish`, `ParseUtil::opc_enum::silent` }
OPC state enum.
- enum `ParseUtil::process_type_enum` {
`ParseUtil::process_type_enum::pt_invalid`, `ParseUtil::process_type_enum::pt_int`, `ParseUtil::process_type_enum::pt_real`,
`ParseUtil::process_type_enum::pt_bool`,
`ParseUtil::process_type_enum::pt_string`, `ParseUtil::process_type_enum::pt_enum`, `ParseUtil::process_type_enum::pt_binary` }
Process type.
- enum `ParseUtil::process_tag_enum` { `ParseUtil::process_tag_enum::all`, `ParseUtil::process_tag_enum::atomic`,
`ParseUtil::process_tag_enum::structured` }
Tag processing enum.
- enum `ParseUtil::process_substitution_enum` { `ParseUtil::process_substitution_enum::standard`, `ParseUtil::process_substitution_enum::ignore` }
Substitution processing enum.

6.6.1 Detailed Description

6.6.2 Typedef Documentation

6.6.2.1 `property_el`

```
using ParseUtil::property_el = typedef std::pair<int, std::stringcase>
```

This pair stores one element of opc properties.

Definition at line 160 of file ParseUtil.h.

6.6.2.2 `property_map`

```
using ParseUtil::property_map = typedef std::map<int, std::stringcase>
```

This map stores a list of opc properties.

Definition at line 156 of file ParseUtil.h.

6.6.2.3 `replacement_table`

```
using ParseUtil::replacement_table = typedef std::map<std::stringcase, std::stringcase>
```

Table of replacement rules

Definition at line 21 of file ParseUtil.h.

6.6.2.4 `substitution_map`

```
using ParseUtil::substitution_map = typedef std::map<std::stringcase, substitution>
```

Map of channel substitution.

Class to specify a map of channel substitutions

Definition at line 619 of file ParseUtil.h.

6.6.3 Enumeration Type Documentation

6.6.3.1 `opc_enum`

```
enum ParseUtil::opc_enum [strong]
```

OPC state enum.

This enum denotes the opc state.

Enumerator

no_change	Do not change inherited behaviour.
publish	Publish.
silent	Do not publish.

Definition at line 144 of file ParseUtil.h.

6.6.3.2 process_substitution_enum

```
enum ParseUtil::process_substitution_enum [strong]
```

Substitution processing enum.

Enumerated type to describe the substitution processing

Enumerator

standard	Standard: ignore if not found in substitution list and not published.
all	Include all.
ignore	Ignore all that are not found.

Definition at line 606 of file ParseUtil.h.

6.6.3.3 process_tag_enum

```
enum ParseUtil::process_tag_enum [strong]
```

Tag processing enum.

Enumerated type to describe the tag processing

Enumerator

all	Process all data types.
atomic	Process atomic data types.
structured	Process structured data type (array, struct, function block)

Definition at line 472 of file ParseUtil.h.

6.6.3.4 process_type_enum

```
enum ParseUtil::process_type_enum [strong]
```

Process type.

Enumerated type to describe the process type

Enumerator

pt_invalid	Invalid type.
pt_int	Numeral type.
pt_real	Floating point type.
pt_bool	Logic type.
pt_string	String type.
pt_enum	Enumerated type.
pt_binary	Binary type.

Definition at line 346 of file ParseUtil.h.

6.7 OPC property constants

Variables

- `constexpr int ParseUtil::OPC_PROP_CDT = 1`
- `constexpr int ParseUtil::OPC_PROP_VALUE = 2`
- `constexpr int ParseUtil::OPC_PROP_QUALITY = 3`
- `constexpr int ParseUtil::OPC_PROP_TIME = 4`
- `constexpr int ParseUtil::OPC_PROP_RIGHTS = 5`
- `constexpr int ParseUtil::OPC_PROP_SCANRATE = 6`
- `constexpr int ParseUtil::OPC_PROP_UNIT = 100`
- `constexpr int ParseUtil::OPC_PROP_DESC = 101`
- `constexpr int ParseUtil::OPC_PROP_HIEU = 102`
- `constexpr int ParseUtil::OPC_PROP_LOEU = 103`
- `constexpr int ParseUtil::OPC_PROP_HIRANGE = 104`
- `constexpr int ParseUtil::OPC_PROP_LORANGE = 105`
- `constexpr int ParseUtil::OPC_PROP_CLOSE = 106`
- `constexpr int ParseUtil::OPC_PROP_OPEN = 107`
- `constexpr int ParseUtil::OPC_PROP_TIMEZONE = 108`
- `constexpr int ParseUtil::OPC_PROP_FGC = 201`
- `constexpr int ParseUtil::OPC_PROP_BGC = 202`
- `constexpr int ParseUtil::OPC_PROP_BLINK = 203`
- `constexpr int ParseUtil::OPC_PROP_BMP = 204`
- `constexpr int ParseUtil::OPC_PROP SND = 205`
- `constexpr int ParseUtil::OPC_PROP_HTML = 206`
- `constexpr int ParseUtil::OPC_PROP_AVI = 207`
- `constexpr int ParseUtil::OPC_PROP_ALMSTAT = 300`
- `constexpr int ParseUtil::OPC_PROP_ALMHELP = 301`
- `constexpr int ParseUtil::OPC_PROP_ALMAREAS = 302`
- `constexpr int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303`
- `constexpr int ParseUtil::OPC_PROP_ALMCONDITION = 304`
- `constexpr int ParseUtil::OPC_PROP_ALMLIMIT = 305`
- `constexpr int ParseUtil::OPC_PROP_ALMDB = 306`
- `constexpr int ParseUtil::OPC_PROP_ALMH = 307`
- `constexpr int ParseUtil::OPC_PROP_ALMH = 308`
- `constexpr int ParseUtil::OPC_PROP_ALML = 309`
- `constexpr int ParseUtil::OPC_PROP_ALMLL = 310`
- `constexpr int ParseUtil::OPC_PROP_ALMROC = 311`
- `constexpr int ParseUtil::OPC_PROP_ALMDEV = 312`
- `constexpr int ParseUtil::OPC_PROP_PREC = 8500`
- `constexpr int ParseUtil::OPC_PROP_ZRST = 8510`
- `constexpr int ParseUtil::OPC_PROP_FFST = 8525`
- `constexpr int ParseUtil::OPC_PROP_RECTYPE = 8600`
- `constexpr int ParseUtil::OPC_PROP_INOUT = 8601`
- `const char *const ParseUtil::OPC_PROP_INPUT = "input"`
- `const char *const ParseUtil::OPC_PROP_OUTPUT = "output"`
- `constexpr int ParseUtil::OPC_PROP_TSE = 8602`
- `constexpr int ParseUtil::OPC_PROP_PINI = 8603`
- `constexpr int ParseUtil::OPC_PROP_DTYP = 8604`
- `constexpr int ParseUtil::OPC_PROP_SERVER = 8610`
- `constexpr int ParseUtil::OPC_PROP_PLNAME = 8611`
- `constexpr int ParseUtil::OPC_PROP_ALIAS = 8620`
- `const char *const ParseUtil::OPC_NAME_ALIAS = "ALIAS"`
- `constexpr int ParseUtil::OPC_PROP_ALMOSV = 8700`

- `constexpr int ParseUtil::OPC_PROP_ALMZSV = 8701`
- `constexpr int ParseUtil::OPC_PROP_ALMCOSV = 8702`
- `constexpr int ParseUtil::OPC_PROP_ALMUNSV = 8703`
- `constexpr int ParseUtil::OPC_PROP_ALMZRSV = 8710`
- `constexpr int ParseUtil::OPC_PROP_ALMFFSV = 8725`
- `constexpr int ParseUtil::OPC_PROP_ALMHHSV = 8727`
- `constexpr int ParseUtil::OPC_PROP_ALMHHSV = 8728`
- `constexpr int ParseUtil::OPC_PROP_ALMLSV = 8729`
- `constexpr int ParseUtil::OPC_PROP_ALMLLSV = 8730`
- `constexpr int ParseUtil::OPC_PROP_FIELD_BEG = 8800`
- `constexpr int ParseUtil::OPC_PROP_FIELD_END = 9000`

6.7.1 Detailed Description

6.7.2 Variable Documentation

6.7.2.1 OPC_NAME_ALIAS

```
const char* const ParseUtil::OPC_NAME_ALIAS = "ALIAS"
```

alias name

Definition at line 70 of file ParseUtilConst.h.

Referenced by ParseUtil::replacement_rules::parse_rules().

6.7.2.2 OPC_PROP_ALIAS

```
constexpr int ParseUtil::OPC_PROP_ALIAS = 8620 [constexpr]
```

alias for structure item or symbol name

Definition at line 69 of file ParseUtilConst.h.

Referenced by ParseUtil::variable_name::append().

6.7.2.3 OPC_PROP_ALMAREAS

```
constexpr int ParseUtil::OPC_PROP_ALMAREAS = 302 [constexpr]
```

area

Definition at line 45 of file ParseUtilConst.h.

6.7.2.4 OPC_PROP_ALMCONDITION

```
constexpr int ParseUtil::OPC_PROP_ALMCONDITION = 304 [constexpr]
```

condition

Definition at line 47 of file ParseUtilConst.h.

6.7.2.5 OPC_PROP_ALMCOSV

```
constexpr int ParseUtil::OPC_PROP_ALMCOSV = 8702 [constexpr]
```

alarm: change of state severity

Definition at line 73 of file ParseUtilConst.h.

6.7.2.6 OPC_PROP_ALMDB

```
constexpr int ParseUtil::OPC_PROP_ALMDB = 306 [constexpr]
```

dead band, tolerance

Definition at line 49 of file ParseUtilConst.h.

6.7.2.7 OPC_PROP_ALMDEV

```
constexpr int ParseUtil::OPC_PROP_ALMDEV = 312 [constexpr]
```

deviation

Definition at line 55 of file ParseUtilConst.h.

6.7.2.8 OPC_PROP_ALMFFSV

```
constexpr int ParseUtil::OPC_PROP_ALMFFSV = 8725 [constexpr]
```

alarm: fifteen state severity

Definition at line 76 of file ParseUtilConst.h.

6.7.2.9 OPC_PROP_ALMH

```
constexpr int ParseUtil::OPC_PROP_ALMH = 308 [constexpr]
```

high alarm -> HIGH

Definition at line 51 of file ParseUtilConst.h.

6.7.2.10 OPC_PROP_ALMHELP

```
constexpr int ParseUtil::OPC_PROP_ALMHELP = 301 [constexpr]
```

help

Definition at line 44 of file ParseUtilConst.h.

6.7.2.11 OPC_PROP_ALMHH

```
constexpr int ParseUtil::OPC_PROP_ALMHH = 307 [constexpr]
```

high high alarm -> HIHI

Definition at line 50 of file ParseUtilConst.h.

6.7.2.12 OPC_PROP_ALMHHSV

```
constexpr int ParseUtil::OPC_PROP_ALMHHSV = 8727 [constexpr]
```

alarm: hihi severity

Definition at line 77 of file ParseUtilConst.h.

6.7.2.13 OPC_PROP_ALMHSV

```
constexpr int ParseUtil::OPC_PROP_ALMHSV = 8728 [constexpr]
```

alarm: high severity

Definition at line 78 of file ParseUtilConst.h.

6.7.2.14 OPC_PROP_ALML

```
constexpr int ParseUtil::OPC_PROP_ALML = 309 [constexpr]
```

low alarm -> LOW

Definition at line 52 of file ParseUtilConst.h.

6.7.2.15 OPC_PROP_ALMLIMIT

```
constexpr int ParseUtil::OPC_PROP_ALMLIMIT = 305 [constexpr]
```

limit

Definition at line 48 of file ParseUtilConst.h.

6.7.2.16 OPC_PROP_ALMILL

```
constexpr int ParseUtil::OPC_PROP_ALMILL = 310 [constexpr]
```

low low alaam -> LOLO

Definition at line 53 of file ParseUtilConst.h.

6.7.2.17 OPC_PROP_ALMILLSV

```
constexpr int ParseUtil::OPC_PROP_ALMILLSV = 8730 [constexpr]
```

alarm: lolo severity

Definition at line 80 of file ParseUtilConst.h.

6.7.2.18 OPC_PROP_ALMLSV

```
constexpr int ParseUtil::OPC_PROP_ALMLSV = 8729 [constexpr]
```

alarm: low severity

Definition at line 79 of file ParseUtilConst.h.

6.7.2.19 OPC_PROP_ALMOSV

```
constexpr int ParseUtil::OPC_PROP_ALMOSV = 8700 [constexpr]
```

alarm: one severity

Definition at line 71 of file ParseUtilConst.h.

6.7.2.20 OPC_PROP_ALMPRIMARYAREA

```
constexpr int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303 [constexpr]
```

primery area

Definition at line 46 of file ParseUtilConst.h.

6.7.2.21 OPC_PROP_ALMROC

```
constexpr int ParseUtil::OPC_PROP_ALMROC = 311 [constexpr]
```

rate of change

Definition at line 54 of file ParseUtilConst.h.

6.7.2.22 OPC_PROP_ALMSTAT

```
constexpr int ParseUtil::OPC_PROP_ALMSTAT = 300 [constexpr]
```

status

Definition at line 43 of file ParseUtilConst.h.

6.7.2.23 OPC_PROP_ALMUNSV

```
constexpr int ParseUtil::OPC_PROP_ALMUNSV = 8703 [constexpr]
```

alarm: unknown state severity

Definition at line 74 of file ParseUtilConst.h.

6.7.2.24 OPC_PROP_ALMZRSV

```
constexpr int ParseUtil::OPC_PROP_ALMZRSV = 8710 [constexpr]
```

alarm: one state severity

Definition at line 75 of file ParseUtilConst.h.

6.7.2.25 OPC_PROP_ALMZSV

```
constexpr int ParseUtil::OPC_PROP_ALMZSV = 8701 [constexpr]
```

alarm: zero severity

Definition at line 72 of file ParseUtilConst.h.

6.7.2.26 OPC_PROP_AVI

```
constexpr int ParseUtil::OPC_PROP_AVI = 207 [constexpr]
```

avi file

Definition at line 41 of file ParseUtilConst.h.

6.7.2.27 OPC_PROP_BGC

```
constexpr int ParseUtil::OPC_PROP_BGC = 202 [constexpr]
```

background color

Definition at line 36 of file ParseUtilConst.h.

6.7.2.28 OPC_PROP_BLINK

```
constexpr int ParseUtil::OPC_PROP_BLINK = 203 [constexpr]
```

blinking

Definition at line 37 of file ParseUtilConst.h.

6.7.2.29 OPC_PROP_BMP

```
constexpr int ParseUtil::OPC_PROP_BMP = 204 [constexpr]
```

bmp file

Definition at line 38 of file ParseUtilConst.h.

6.7.2.30 OPC_PROP_CDT

```
constexpr int ParseUtil::OPC_PROP_CDT = 1 [constexpr]
```

canonocal data type

Definition at line 18 of file ParseUtilConst.h.

6.7.2.31 OPC_PROP_CLOSE

```
constexpr int ParseUtil::OPC_PROP_CLOSE = 106 [constexpr]
```

label for close state -> ONAM

Definition at line 31 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.7.2.32 OPC_PROP_DESC

```
constexpr int ParseUtil::OPC_PROP_DESC = 101 [constexpr]
```

description string -> DESC

Definition at line 26 of file ParseUtilConst.h.

6.7.2.33 OPC_PROP_DTYP

```
constexpr int ParseUtil::OPC_PROP_DTYP = 8604 [constexpr]
```

DTYP field: opc or opcRaw

Definition at line 66 of file ParseUtilConst.h.

6.7.2.34 OPC_PROP_FFST

```
constexpr int ParseUtil::OPC_PROP_FFST = 8525 [constexpr]
```

... fifteen string

Definition at line 59 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.7.2.35 OPC_PROP_FGC

```
constexpr int ParseUtil::OPC_PROP_FGC = 201 [constexpr]
```

foreground color

Definition at line 35 of file ParseUtilConst.h.

6.7.2.36 OPC_PROP_FIELD_BEG

```
constexpr int ParseUtil::OPC_PROP_FIELD_BEG = 8800 [constexpr]
```

Beginning of field,value combination

Definition at line 81 of file ParseUtilConst.h.

6.7.2.37 OPC_PROP_FIELD_END

```
constexpr int ParseUtil::OPC_PROP_FIELD_END = 9000 [constexpr]
```

End of field,value combinations

Definition at line 82 of file ParseUtilConst.h.

6.7.2.38 OPC_PROP_HIEU

```
constexpr int ParseUtil::OPC_PROP_HIEU = 102 [constexpr]
```

high expectation value -> HOPR

Definition at line 27 of file ParseUtilConst.h.

6.7.2.39 OPC_PROP_HIRANGE

```
constexpr int ParseUtil::OPC_PROP_HIRANGE = 104 [constexpr]
```

absolute maximum value -> DRVH

Definition at line 29 of file ParseUtilConst.h.

6.7.2.40 OPC_PROP_HTML

```
constexpr int ParseUtil::OPC_PROP_HTML = 206 [constexpr]
```

html file

Definition at line 40 of file ParseUtilConst.h.

6.7.2.41 OPC_PROP_INOUT

```
constexpr int ParseUtil::OPC_PROP_INOUT = 8601 [constexpr]
```

input or output

Definition at line 61 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.42 OPC_PROP_INPUT

```
const char* const ParseUtil::OPC_PROP_INPUT = "input"
```

input

Definition at line 62 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.43 OPC_PROP_LOEU

```
constexpr int ParseUtil::OPC_PROP_LOEU = 103 [constexpr]
```

low expectation value -> LOPR

Definition at line 28 of file ParseUtilConst.h.

6.7.2.44 OPC_PROP_LORANGE

```
constexpr int ParseUtil::OPC_PROP_LORANGE = 105 [constexpr]
```

absolute minimum value -> DRVL

Definition at line 30 of file ParseUtilConst.h.

6.7.2.45 OPC_PROP_OPEN

```
constexpr int ParseUtil::OPC_PROP_OPEN = 107 [constexpr]
```

label for open state -> ZNAM

Definition at line 32 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.7.2.46 OPC_PROP_OUTPUT

```
const char* const ParseUtil::OPC_PROP_OUTPUT = "output"
```

output

Definition at line 63 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.47 OPC_PROP_PINI

```
constexpr int ParseUtil::OPC_PROP_PINI = 8603 [constexpr]
```

initialization

Definition at line 65 of file ParseUtilConst.h.

6.7.2.48 OPC_PROP_PLNAME

```
constexpr int ParseUtil::OPC_PROP_PLNAME = 8611 [constexpr]
```

tc name including ads routing info and port

Definition at line 68 of file ParseUtilConst.h.

Referenced by ParseUtil::process_arg_tc::get_full(), and ParseTpy::tpy_file::parse_finish().

6.7.2.49 OPC_PROP_PREC

```
constexpr int ParseUtil::OPC_PROP_PREC = 8500 [constexpr]
```

precision

Definition at line 57 of file ParseUtilConst.h.

6.7.2.50 OPC_PROP_QUALITY

```
constexpr int ParseUtil::OPC_PROP_QUALITY = 3 [constexpr]
```

data quality flag

Definition at line 20 of file ParseUtilConst.h.

6.7.2.51 OPC_PROP_RECTYPE

```
constexpr int ParseUtil::OPC_PROP_RECTYPE = 8600 [constexpr]
```

record type

Definition at line 60 of file ParseUtilConst.h.

6.7.2.52 OPC_PROP_RIGHTS

```
constexpr int ParseUtil::OPC_PROP_RIGHTS = 5 [constexpr]
```

access right: 1 read, 2 write, 3 read/write

Definition at line 22 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.7.2.53 OPC_PROP_SCANRATE

```
constexpr int ParseUtil::OPC_PROP_SCANRATE = 6 [constexpr]
```

scan rate

Definition at line 23 of file ParseUtilConst.h.

6.7.2.54 OPC_PROP_SERVER

```
constexpr int ParseUtil::OPC_PROP_SERVER = 8610 [constexpr]
```

server name

Definition at line 67 of file ParseUtilConst.h.

6.7.2.55 OPC_PROP SND

```
constexpr int ParseUtil::OPC_PROP SND = 205 [constexpr]
```

sound file

Definition at line 39 of file ParseUtilConst.h.

6.7.2.56 OPC_PROP_TIME

```
constexpr int ParseUtil::OPC_PROP_TIME = 4 [constexpr]
```

timestamp

Definition at line 21 of file ParseUtilConst.h.

6.7.2.57 OPC_PROP_TIMEZONE

```
constexpr int ParseUtil::OPC_PROP_TIMEZONE = 108 [constexpr]
```

time zone

Definition at line 33 of file ParseUtilConst.h.

6.7.2.58 OPC_PROP_TSE

```
constexpr int ParseUtil::OPC_PROP_TSE = 8602 [constexpr]
```

time stamp

Definition at line 64 of file ParseUtilConst.h.

6.7.2.59 OPC_PROP_UNIT

```
constexpr int ParseUtil::OPC_PROP_UNIT = 100 [constexpr]
```

unit string -> EGU

Definition at line 25 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.7.2.60 OPC_PROP_VALUE

```
constexpr int ParseUtil::OPC_PROP_VALUE = 2 [constexpr]
```

value

Definition at line 19 of file ParseUtilConst.h.

6.7.2.61 OPC_PROP_ZRST

```
constexpr int ParseUtil::OPC_PROP_ZRST = 8510 [constexpr]
```

zero string ...

Definition at line 58 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.8 String functions and classes

Classes

- struct `std::case_char_traits`
case insensitive traits.
- struct `std::case_wchar_traits`
case insensitive unicode traits.

Typedefs

- using `std::stringcase = std::basic_string< char, case_char_traits >`
case insensitive string.
- using `std::wstringcase = std::basic_string< wchar_t, case_wchar_traits >`
case insensitive string.

Functions

- int `std::strncasecmp (const char *s1, const char *s2, size_t n) noexcept`
case insensitive compare with maximum length
- int `std::wcscasewcmp (const wchar_t *s1, const wchar_t *s2, size_t n) noexcept`
case insensitive unicode compare with maximum length
- static bool `std::case_char_traits::eq (const char_type &c1, const char_type &c2) noexcept`
- static bool `std::case_char_traits::ne (const char_type &c1, const char_type &c2) noexcept`
- static bool `std::case_char_traits::lt (const char_type &c1, const char_type &c2) noexcept`
- static int `std::case_char_traits::compare (const char_type *s1, const char_type *s2, size_t n) noexcept`
- static bool `std::case_wchar_traits::eq (const char_type &c1, const char_type &c2) noexcept`
- static bool `std::case_wchar_traits::ne (const char_type &c1, const char_type &c2) noexcept`
- static bool `std::case_wchar_traits::lt (const char_type &c1, const char_type &c2) noexcept`
- static int `std::case_wchar_traits::compare (const char_type *s1, const char_type *s2, size_t n) noexcept`
- void `std::trim_space (std::stringcase &s)`
- void `std::trim_space (std::wstringcase &s)`
- template<class Container , class String , class Predicate >
void `std::split_string (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)`
Splits a strings.

6.8.1 Detailed Description

6.8.2 Typedef Documentation

6.8.2.1 `stringcase`

```
using std::stringcase = typedef std::basic_string <char, case_char_traits>  

case insensitive string.
```

This string class is not case sensitive.

Definition at line 104 of file `stringcase.h`.

6.8.2.2 wstringcase

```
using std::wstringcase = typedef std::basic_string <wchar_t, case_wchar_traits>
```

case insensitive string.

This string class is not case sensitive.

Definition at line 109 of file stringcase.h.

6.8.3 Function Documentation

6.8.3.1 compare() [1/2]

```
static int std::case_char_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static], [noexcept]
```

Compare strings

Parameters

<i>s1</i>	First string
<i>s2</i>	Second string
<i>n</i>	number of characters

Definition at line 64 of file stringcase.h.

References std::strncasecmp().

Here is the call graph for this function:

6.8.3.2 compare() [2/2]

```
static int std::case_wchar_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static], [noexcept]
```

Compare strings

Parameters

<i>s1</i>	First string
<i>s2</i>	Second string
<i>n</i>	number of characters

Definition at line 96 of file stringcase.h.

References std::wcsncasewcmp().

Here is the call graph for this function:

6.8.3.3 eq() [1/2]

```
static bool std::case_char_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static], [noexcept]
```

Equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 45 of file stringcase.h.

6.8.3.4 eq() [2/2]

```
static bool std::case_wchar_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static], [noexcept]
```

Equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 77 of file stringcase.h.

6.8.3.5 lt() [1/2]

```
static bool std::case_char_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static], [noexcept]
```

Lower than character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 57 of file stringcase.h.

6.8.3.6 lt() [2/2]

```
static bool std::case_wchar_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static], [noexcept]
```

Lower than character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 89 of file stringcase.h.

6.8.3.7 ne() [1/2]

```
static bool std::case_char_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static], [noexcept]
```

Not equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 51 of file stringcase.h.

6.8.3.8 ne() [2/2]

```
static bool std::case_wchar_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static], [noexcept]
```

Not equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 83 of file stringcase.h.

6.8.3.9 `split_string()`

```
template<class Container , class String , class Predicate >
void std::split_string (
    Container & output,
    const String & input,
    const Predicate & pred,
    bool trimEmpty = true )
```

Splits a strings.

Splits a string into its tokens and adds them to a container. The delimiter can be easily specified with a lambda expression. Example: `stringcase arg ("This is a test!"); vector<stringcase> list; split_string (list, arg, [] (char c)->bool { return isspace (c) != 0; }, true);`

Parameters

<i>output</i>	Output container
<i>input</i>	Input string
<i>pred</i>	Function which returns true when character is a separator
<i>trimEmpty</i>	Trims empty strings when true

Definition at line 135 of file stringcase.h.

Referenced by `ParseUtil::optarg::parse()`.

Here is the caller graph for this function:



6.8.3.10 strncasecmp()

```
int std::strncasecmp (
    const char * s1,
    const char * s2,
    size_t n ) [inline], [noexcept]
```

case insensitive compare with maximum length

Case insensitive compare.

Parameters

<i>s1</i>	first string
<i>s2</i>	second string
<i>n</i>	Number of characters

Returns

<0 smaller, 0 equal, >0 greater

Definition at line 23 of file stringcase.h.

Referenced by std::case_char_traits::compare().

Here is the caller graph for this function:

6.8.3.11 trim_space() [1/2]

```
void std::trim_space (
    std::stringcase & s )
```

trim space on both ends.

Parameters

<i>s</i>	string to trim
----------	----------------

Referenced by ParseUtil::variable_name::append(), ParseUtil::replacement_rules::apply_replacement_rules(), EpicsTpy::epics_list_processing::operator()(), and ParseUtil::replacement_rules::parse_rules().

Here is the caller graph for this function:

6.8.3.12 trim_space() [2/2]

```
void std::trim_space (
    std::wstringcase & s )
```

trim space on both ends.

Parameters

<i>s</i>	string to trim
----------	----------------

6.8.3.13 wcsncasewcmp()

```
int std::wcsncasewcmp (
    const wchar_t * s1,
    const wchar_t * s2,
    size_t n ) [inline], [noexcept]
```

case insensitive unicode compare with maximum length

Case insensitive compare for unicode string.

Parameters

<i>s1</i>	first string
<i>s2</i>	second string
<i>n</i>	Number of characters

Returns

<0 smaller, 0 equal, >0 greater

Definition at line 34 of file stringcase.h.

Referenced by std::case_wchar_traits::compare().

Here is the caller graph for this function:

6.9 TwinCAT read/write scanning

Classes

- struct `TcComms::DataPar`
Memory location struct.
- class `TcComms::TCatInterface`
TCat interface class.
- class `TcComms::tcProcWrite`
TwinCAT process write requests.
- class `TcComms::TcPLC`
TwinCAT PLC.
- class `TcComms::AmsRouterNotification`
AMS Router Notification.

Variables

- constexpr int `TcComms::MAX_REQ_SIZE` = 250000
maximum allowed request size (bytes)
- constexpr int `TcComms::MAX_SINGLE_GAP_SIZE` = 50
maximum allowed size (bytes) of a memory gap within continuous request
- constexpr double `TcComms::MAX_REL_GAP` = 0.25
(maximum allowed total gap size) / (current request size)
- constexpr int `TcComms::MIN_REL_GAP_SIZE` = 100
minimum allowed relative gap size (bytes)
- constexpr int `TcComms::default_scanrate` = 100
default PLC TwinCAT scan rate (100ms)
- constexpr int `TcComms::minimum_scanrate` = 5
minimum PLC TwinCAT scan rate (5ms)
- constexpr int `TcComms::maximum_scanrate` = 10000
maximum PLC TwinCAT scan rate (10s)
- constexpr int `TcComms::default_multiple` = 10
default multiple for PLC EPICS scan rate (10)
- constexpr int `TcComms::minimum_multiple` = 1
minimum multiple for PLC EPICS scan rate (1)
- constexpr int `TcComms::maximum_multiple` = 200
maximum multiple for PLC EPICS scan rate (200)

6.9.1 Detailed Description

6.10 EPICS utility functions and classes

Classes

- class [EpicsTpy::epics_conversion](#)
Epics conversion.
- class [EpicsTpy::split_io_support](#)
Split IO support.
- class [EpicsTpy::multi_io_support](#)
Multiple IO support.
- class [EpicsTpy::epics_list_processing](#)
List processing.
- struct [EpicsTpy::macro_info](#)
Macro information.
- struct [EpicsTpy::macro_record](#)
Macro record.
- class [EpicsTpy::epics_macrofiles_processing](#)
Macro file processing.
- class [EpicsTpy::epics_db_processing](#)
Epics database record processing

Typedefs

- using [EpicsTpy::macro_list](#) = std::vector< [macro_info](#) >
- using [EpicsTpy::macro_stack](#) = std::stack< [macro_record](#) >
- using [EpicsTpy::filename_set](#) = std::unordered_set< [std::stringcase](#) >

Enumerations

- enum [EpicsTpy::tc_epics_conv](#) { [EpicsTpy::tc_epics_conv::no_conversion](#), [EpicsTpy::tc_epics_conv::no_dot](#), [EpicsTpy::tc_epics_conv::ligo_std](#), [EpicsTpy::tc_epics_conv::ligo_vac](#) }
Conversion rules for TC/EPICS.
- enum [EpicsTpy::case_type](#) { [EpicsTpy::case_type::preserve](#), [EpicsTpy::case_type::upper](#), [EpicsTpy::case_type::lower](#) }
Case conversion rule enum.
- enum [EpicsTpy::io_filestat](#) { [EpicsTpy::io_filestat::closed](#), [read](#), [write](#) }
enum for file io
- enum [EpicsTpy::listing_type](#) { [EpicsTpy::listing_type::standard](#), [EpicsTpy::listing_type::autoburt](#), [EpicsTpy::listing_type::daqini](#) }
Listing type enum.
- enum [EpicsTpy::macrofile_type](#) { [EpicsTpy::macrofile_type::all](#), [EpicsTpy::macrofile_type::fields](#), [EpicsTpy::macrofile_type::error](#) }
Device support enum.
- enum [EpicsTpy::device_support_type](#) { [EpicsTpy::device_support_type::opc_name](#), [EpicsTpy::device_support_type::tc_name](#) }
String support enum.
- enum [EpicsTpy::string_support_type](#) { [EpicsTpy::string_support_type::short_string](#), [EpicsTpy::string_support_type::long_string](#), [vary_string](#) }
String support enum.
- enum [EpicsTpy::int_support_type](#) { [EpicsTpy::int_support_type::int_32](#), [EpicsTpy::int_support_type::int_64](#), [int_auto](#) }
Integer support enum.

6.10.1 Detailed Description

6.10.2 Typedef Documentation

6.10.2.1 filename_set

```
using EpicsTpy::filename_set = typedef std::unordered_set<std::stringcase>
```

A set of filenames

Definition at line 564 of file TpyToEpics.h.

6.10.2.2 macro_list

```
using EpicsTpy::macro_list = typedef std::vector<macro_info>
```

A list of fields

Definition at line 537 of file TpyToEpics.h.

6.10.2.3 macro_stack

```
using EpicsTpy::macro_stack = typedef std::stack<macro_record>
```

A stack of records/structs

Definition at line 560 of file TpyToEpics.h.

6.10.3 Enumeration Type Documentation

6.10.3.1 case_type

```
enum EpicsTpy::case_type [strong]
```

Case conversion rule enum.

This enum describes the case conversion rule

Enumerator

preserve	Preserve the case.
upper	Convert to upper case.
lower	Convert to lower case.

Definition at line 46 of file TpyToEpics.h.

6.10.3.2 device_support_type

```
enum EpicsTpy::device_support_type [strong]
```

Device support enum.

This enum describes the type of device support to use

Enumerator

opc_name	Use opc names in the INPUT/OUTPUT epics fields.
tc_name	Use TwinCAT names in the INPUT/OUTPUT epics fields.

Definition at line 688 of file TpyToEpics.h.

6.10.3.3 int_support_type

```
enum EpicsTpy::int_support_type [strong]
```

Integer support enum.

This enum describes the type of int record to use

Enumerator

int_32	Use standard 32-bit integers (longin/longout)
int_64	Use 64-bit integers (int64in/int64out)

Definition at line 710 of file TpyToEpics.h.

6.10.3.4 io_filestat

```
enum EpicsTpy::io_filestat [strong]
```

enum for file io

Enumerator

closed	file is closed
--------	----------------

Definition at line 312 of file TpyToEpics.h.

6.10.3.5 listing_type

```
enum EpicsTpy::listing_type [strong]
```

Listing type enum.

This enum describes the type of listing to produce

Enumerator

standard	Standard listing using TwinCAT/OPC names.
autoburt	Autoburt listing.
daqini	LIGO DAQ ini listing.

Definition at line 419 of file TpyToEpics.h.

6.10.3.6 macrofile_type

```
enum EpicsTpy::macrofile_type [strong]
```

This enum describes the type of macros to produce

Enumerator

all	Include all fields and error messages.
fields	Include all fields.
errors	Include error messages.

Definition at line 509 of file TpyToEpics.h.

6.10.3.7 string_support_type

```
enum EpicsTpy::string_support_type [strong]
```

String support enum.

This enum describes the type of string record to use

Enumerator

short_string	Use standard strings (stringin/stringout)
long_string	Use long strings (lqi/lso)

Definition at line 698 of file TpyToEpics.h.

6.10.3.8 tc_epics_conv

```
enum EpicsTpy::tc_epics_conv [strong]
```

Conversion rules for TC/EPICS.

This enum describes the TwinCAT/opc to EPICS conversion rule

Enumerator

no_conversion	No conversion.
no_dot	Convert '.' to '_'.
ligo_std	LIGO standard conversion: Eliminate leading '.', Replace second '.' with ':', Replace third '.' with '-', Replace all other '.' with '_'
ligo_vac	LIGO standard conversion for vacuum channels: Eliminate leading '.', Replace second '.' with ':', Replace third '.' with '_', Replace all other '.' with '_'

Definition at line 24 of file TpyToEpics.h.

6.11 EPICS constants

Variables

- `constexpr int EpicsTpy::MAX_EPICS_CHANNEL = 54`
- `constexpr int EpicsTpy::MAX_EPICS_DESC = 40`
- `constexpr int EpicsTpy::MAX_EPICS_STRING = 40`
- `constexpr int EpicsTpy::MAX_EPICS_LONGSTRING = 65535`
- `constexpr int EpicsTpy::MAX_EPICS_UNIT = 15`
- `constexpr int EpicsTpy::MAX_EPICS_ENUM = 25`
- `const char *const EpicsTpy::EPICS_DB_EGU = "EGU"`
- `const char *const EpicsTpy::EPICS_DB_DESC = "DESC"`
- `const char *const EpicsTpy::EPICS_DB_HOPR = "HOPR"`
- `const char *const EpicsTpy::EPICS_DB_LOPR = "LOPR"`
- `const char *const EpicsTpy::EPICS_DB_DRVH = "DRVH"`
- `const char *const EpicsTpy::EPICS_DB_DRVL = "DRVL"`
- `const char *const EpicsTpy::EPICS_DB_ONAM = "ONAM"`
- `const char *const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"`
- `const char *const EpicsTpy::EPICS_DB_PREC = "PREC"`
- `const char *const EpicsTpy::EPICS_DB_ZRST [16]`
- `const char *const EpicsTpy::EPICS_DB_ZRVL [16]`
- `const char *const EpicsTpy::EPICS_DB_SCAN = "SCAN"`
- `const char *const EpicsTpy::EPICS_DB_INP = "INP"`
- `const char *const EpicsTpy::EPICS_DB_OUT = "OUT"`
- `const char *const EpicsTpy::EPICS_DB_TSE = "TSE"`
- `const char *const EpicsTpy::EPICS_DB_PINI = "PINI"`
- `const char *const EpicsTpy::EPICS_DB_DTYP = "DTYP"`
- `const char *const EpicsTpy::EPICS_DB_SIZV = "SIZV"`
- `const char *const EpicsTpy::EPICS_DB_OSV = "OSV"`
- `const char *const EpicsTpy::EPICS_DB_ZSV = "ZSV"`
- `const char *const EpicsTpy::EPICS_DB_COSV = "COSV"`
- `const char *const EpicsTpy::EPICS_DB_HIHI = "HIHI"`
- `const char *const EpicsTpy::EPICS_DB_HIGH = "HIGH"`
- `const char *const EpicsTpy::EPICS_DB_LOW = "LOW"`
- `const char *const EpicsTpy::EPICS_DB_LOLO = "LOLO"`
- `const char *const EpicsTpy::EPICS_DB_HYST = "HYST"`
- `const char *const EpicsTpy::EPICS_DB_HHSV = "HHSV"`
- `const char *const EpicsTpy::EPICS_DB_HSV = "HSV"`
- `const char *const EpicsTpy::EPICS_DB_LSV = "LSV"`
- `const char *const EpicsTpy::EPICS_DB_LLSV = "LLSV"`
- `const char *const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"`
- `const char *const EpicsTpy::EPICS_DB_MINOR = "MINOR"`
- `const char *const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"`
- `const char *const EpicsTpy::EPICS_DB_UNSV = "UNSV"`
- `const char *const EpicsTpy::EPICS_DB_ZRSV [16]`
- `const char *const EpicsTpy::EPICS_DB_FORBIDDEN []`
- `const char *const EpicsTpy::EPICS_DB_ALLOWED []`
- `const char *const EpicsTpy::EPICS_DB_NUMVAL []`
- `const char *const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"`
- `constexpr int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4`
- `constexpr int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2`
- `constexpr int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT`
- `const char *const EpicsTpy::LIGODAQ_UNIT_NAME = "units"`
- `const char *const EpicsTpy::LIGODAQ_UNIT_NONE = "none"`
- `const char *const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE`
- `const char *const EpicsTpy::LIGODAQ_INI_HEADER`

6.11.1 Detailed Description

6.11.2 Variable Documentation

6.11.2.1 EPICS_DB_ALLOWED

```
const char* const EpicsTpy::EPICS_DB_ALLOWED[ ]
```

Initial value:

```
= {"LINR", "EGUF", "EGUL", "AOFF", "ASLO", "ESLO", "SMOO", "HIHI", "LOLO", "HIGH", "LOW", "HHSV", "LLSV",
     "HSV", "LSV", "HYST", "ZSV", "OSV", "COSV", "IVOA", "IVOV", "UNSV", "ZRSV",
     "ONSV", "TWSV", "THSV",
     "FRSV", "FVSV", "SXSV", "SVSV", "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV",
     "FTSV", "FFSV", NULL}
```

Names of allowed EPICS record fields

Definition at line 78 of file TpyToEpicsConst.h.

6.11.2.2 EPICS_DB_COSV

```
const char* const EpicsTpy::EPICS_DB_COSV = "COSV"
```

change severity

Definition at line 50 of file TpyToEpicsConst.h.

6.11.2.3 EPICS_DB_DESC

```
const char* const EpicsTpy::EPICS_DB_DESC = "DESC"
```

description string

Definition at line 26 of file TpyToEpicsConst.h.

6.11.2.4 EPICS_DB_DRVH

```
const char* const EpicsTpy::EPICS_DB_DRVH = "DRVH"
```

drive high string

Definition at line 29 of file TpyToEpicsConst.h.

6.11.2.5 EPICS_DB_DRVL

```
const char* const EpicsTpy::EPICS_DB_DRVL = "DRVL"
```

drive low string

Definition at line 30 of file TpyToEpicsConst.h.

6.11.2.6 EPICS_DB_DTYP

```
const char* const EpicsTpy::EPICS_DB_DTYP = "DTYP"
```

data type

Definition at line 45 of file TpyToEpicsConst.h.

6.11.2.7 EPICS_DB_EGU

```
const char* const EpicsTpy::EPICS_DB_EGU = "EGU"
```

unit string

Definition at line 25 of file TpyToEpicsConst.h.

6.11.2.8 EPICS_DB_FORBIDDEN

```
const char* const EpicsTpy::EPICS_DB_FORBIDDEN[ ]
```

Initial value:

```
= {"", EPICS_DB_SCAN, "DOL", EPICS_DB_INP, EPICS_DB_OUT, "VAL", "RVAL", "INIT",
     "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
     "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of forbidden EPICS record fields

Definition at line 72 of file TpyToEpicsConst.h.

6.11.2.9 EPICS_DB_HHSV

```
const char* const EpicsTpy::EPICS_DB_HHSV = "HHSV"
```

high severity high limit

Definition at line 57 of file TpyToEpicsConst.h.

6.11.2.10 EPICS_DB_HIGH

```
const char* const EpicsTpy::EPICS_DB_HIGH = "HIGH"
```

low severity high limit

Definition at line 53 of file TpyToEpicsConst.h.

6.11.2.11 EPICS_DB_HIHI

```
const char* const EpicsTpy::EPICS_DB_HIHI = "HIHI"
```

high severity high limit

Definition at line 52 of file TpyToEpicsConst.h.

6.11.2.12 EPICS_DB_HOPR

```
const char* const EpicsTpy::EPICS_DB_HOPR = "HOPR"
```

high ops value string

Definition at line 27 of file TpyToEpicsConst.h.

6.11.2.13 EPICS_DB_HSV

```
const char* const EpicsTpy::EPICS_DB_HSV = "HSV"
```

low severity high limit

Definition at line 58 of file TpyToEpicsConst.h.

6.11.2.14 EPICS_DB_HYST

```
const char* const EpicsTpy::EPICS_DB_HYST = "HYST"
```

deadband

Definition at line 56 of file TpyToEpicsConst.h.

6.11.2.15 EPICS_DB_INP

```
const char* const EpicsTpy::EPICS_DB_INP = "INP"
```

input link

Definition at line 41 of file TpyToEpicsConst.h.

6.11.2.16 EPICS_DB_LLSV

```
const char* const EpicsTpy::EPICS_DB_LLSV = "LLSV"
```

high severity low limit

Definition at line 60 of file TpyToEpicsConst.h.

6.11.2.17 EPICS_DB_LOLO

```
const char* const EpicsTpy::EPICS_DB_LOLO = "LOLO"
```

high severity low limit

Definition at line 55 of file TpyToEpicsConst.h.

6.11.2.18 EPICS_DB_LOPR

```
const char* const EpicsTpy::EPICS_DB_LOPR = "LOPR"
```

low ops value string

Definition at line 28 of file TpyToEpicsConst.h.

6.11.2.19 EPICS_DB_LOW

```
const char* const EpicsTpy::EPICS_DB_LOW = "LOW"
```

low severity low limit

Definition at line 54 of file TpyToEpicsConst.h.

6.11.2.20 EPICS_DB_LSV

```
const char* const EpicsTpy::EPICS_DB_LSV = "LSV"
```

low severity low limit

Definition at line 59 of file TpyToEpicsConst.h.

6.11.2.21 EPICS_DB_MAJOR

```
const char* const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
```

major alarm

Definition at line 64 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::process_field_alarm().

6.11.2.22 EPICS_DB_MINOR

```
const char* const EpicsTpy::EPICS_DB_MINOR = "MINOR"
```

minor alarm

Definition at line 63 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::process_field_alarm().

6.11.2.23 EPICS_DB_NOALARM

```
const char* const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
```

no alarm

Definition at line 62 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::process_field_alarm().

6.11.2.24 EPICS_DB_NUMVAL

```
const char* const EpicsTpy::EPICS_DB_NUMVAL[ ]
```

Initial value:

```
= {EPICS_DB_TSE, EPICS_DB_HOPR, EPICS_DB_LOPR, EPICS_DB_DRVH, EPICS_DB_DRVL, "EGUF", "EGUL",
    "AOFF", "ASLO", "ESLO", "SMOO", "PREC", "HIHI", "LOLO", "HIGH", "LOW", "HYST",
    "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL", "EIVL", "NIVL",
    "TEVL", "ELVL",
    "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of EPICS record fields which are numeric

Definition at line 84 of file TpyToEpicsConst.h.

6.11.2.25 EPICS_DB_ONAM

```
const char* const EpicsTpy::EPICS_DB_ONAM = "ONAM"
```

one name string

Definition at line 31 of file TpyToEpicsConst.h.

6.11.2.26 EPICS_DB_OSV

```
const char* const EpicsTpy::EPICS_DB_OSV = "OSV"
```

one severity

Definition at line 48 of file TpyToEpicsConst.h.

6.11.2.27 EPICS_DB_OUT

```
const char* const EpicsTpy::EPICS_DB_OUT = "OUT"
```

output link

Definition at line 42 of file TpyToEpicsConst.h.

6.11.2.28 EPICS_DB_PINI

```
const char* const EpicsTpy::EPICS_DB_PINI = "PINI"
```

initialization

Definition at line 44 of file TpyToEpicsConst.h.

6.11.2.29 EPICS_DB_PREC

```
const char* const EpicsTpy::EPICS_DB_PREC = "PREC"
```

precision

Definition at line 34 of file TpyToEpicsConst.h.

6.11.2.30 EPICS_DB_SCAN

```
const char* const EpicsTpy::EPICS_DB_SCAN = "SCAN"
```

Scan

Definition at line 40 of file TpyToEpicsConst.h.

6.11.2.31 EPICS_DB_SIZV

```
const char* const EpicsTpy::EPICS_DB_SIZV = "SIZV"
```

string size for long strings

Definition at line 46 of file TpyToEpicsConst.h.

6.11.2.32 EPICS_DB_TSE

```
const char* const EpicsTpy::EPICS_DB_TSE = "TSE"
```

time stamp

Definition at line 43 of file TpyToEpicsConst.h.

6.11.2.33 EPICS_DB_UNSV

```
const char* const EpicsTpy::EPICS_DB_UNSV = "UNSV"
```

unknown severity

Definition at line 66 of file TpyToEpicsConst.h.

6.11.2.34 EPICS_DB_ZNAM

```
const char* const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"
```

zero name string

Definition at line 32 of file TpyToEpicsConst.h.

6.11.2.35 EPICS_DB_ZRST

```
const char* const EpicsTpy::EPICS_DB_ZRST[16]
```

Initial value:

```
= {"ZRST", "ONST", "TWST", "THST", "FRST", "FVST", "SXST", "SVST",
    "EIST", "NIST", "TEST", "ELST", "TVST", "TTST", "FTST", "FFST"}
```

enum string

Definition at line 35 of file TpyToEpicsConst.h.

6.11.2.36 EPICS_DB_ZRSV

```
const char* const EpicsTpy::EPICS_DB_ZRSV[16]
```

Initial value:

```
= {"ZRSV", "ONSV", "TWSV", "THSV", "FRSV", "FVSV", "SXSV", "SVSV",
    "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV", "FTSV", "FFSV"}
```

enum alarm severity

Definition at line 67 of file TpyToEpicsConst.h.

6.11.2.37 EPICS_DB_ZRVL

```
const char* const EpicsTpy::EPICS_DB_ZRVL[16]
```

Initial value:

```
= {"ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
    "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL"}
```

enum val

Definition at line 37 of file TpyToEpicsConst.h.

6.11.2.38 EPICS_DB_ZSV

```
const char* const EpicsTpy::EPICS_DB_ZSV = "ZSV"
```

zero severity

Definition at line 49 of file TpyToEpicsConst.h.

6.11.2.39 LIGODAQ_DATATYPE_DEFAULT

```
constexpr int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT [constexpr]
```

default DAQ datatype is float

Definition at line 94 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.40 LIGODAQ_DATATYPE_FLOAT

```
constexpr int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4 [constexpr]
```

DAQ datatype is float

Definition at line 92 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.41 LIGODAQ_DATATYPE_INT32

```
constexpr int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2 [constexpr]
```

DAQ datatype is int32

Definition at line 93 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.42 LIGODAQ_DATATYPE_NAME

```
const char* const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"
```

DAQ datatype name

Definition at line 91 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.43 LIGODAQ_INI_HEADER

```
const char* const EpicsTpy::LIGODAQ_INI_HEADER
```

Initial value:

```
=  
"[default]\n"  
"gain=1.00\n"  
"datatype=%i\n"  
"ifoid=0\n"  
"slope=6.1028e-05\n"  
"acquire=3\n"  
"offset=0\n"  
"units=%s\n"  
"dcuid=4\n"  
"datarate=16"
```

DAQ ini file header: substitute defaults

Definition at line 101 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.44 LIGODAQ_UNIT_DEFAULT

```
const char* const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
```

default DAQ unit is none

Definition at line 98 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.45 LIGODAQ_UNIT_NAME

```
const char* const EpicsTpy::LIGODAQ_UNIT_NAME = "units"
```

DAQ unit name

Definition at line 96 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.46 LIGODAQ_UNIT_NONE

```
const char* const EpicsTpy::LIGODAQ_UNIT_NONE = "none"
```

DAQ unit for no unit

Definition at line 97 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.11.2.47 MAX_EPICS_CHANNEL

```
constexpr int EpicsTpy::MAX_EPICS_CHANNEL = 54 [constexpr]
```

maximum length of EPICS channel name

Definition at line 18 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.11.2.48 MAX_EPICS_DESC

```
constexpr int EpicsTpy::MAX_EPICS_DESC = 40 [constexpr]
```

maximum length of EPICS channel description

Definition at line 19 of file TpyToEpicsConst.h.

6.11.2.49 MAX_EPICS_ENUM

```
constexpr int EpicsTpy::MAX_EPICS_ENUM = 25 [constexpr]
```

maximum length of EPICS channel enum string

Definition at line 23 of file TpyToEpicsConst.h.

6.11.2.50 MAX_EPICS_LONGSTRING

```
constexpr int EpicsTpy::MAX_EPICS_LONGSTRING = 65535 [constexpr]
```

maximum length of EPICS long strings

Definition at line 21 of file TpyToEpicsConst.h.

6.11.2.51 MAX_EPICS_STRING

```
constexpr int EpicsTpy::MAX_EPICS_STRING = 40 [constexpr]
```

maximum length of EPICS strings

Definition at line 20 of file TpyToEpicsConst.h.

6.11.2.52 MAX_EPICS_UNIT

```
constexpr int EpicsTpy::MAX_EPICS_UNIT = 15 [constexpr]
```

maximum length of EPICS channel unit string

Definition at line 22 of file TpyToEpicsConst.h.

Chapter 7

Namespace Documentation

7.1 DevTc Namespace Reference

Namespace for TCat device support.

Classes

- struct `devTcDefIn`
Device support input record.
- struct `devTcDefLo`
Device support record.
- struct `devTcDefOut`
device support output record.
- struct `devTcDefWaveformIn`
device support waveform record.
- struct `epics_record_traits`
Epics record traits.
- class `epics_tc_db_processing`
EPICS/TCat db processing.
- class `EpicsInterface`
Epics interface class.
- class `register_devsup`
Device support registration.
- class `tcRegisterTolocShell`
Register TC commands.

Typedefs

- using `filename_rule_list_tuple` = std::tuple<`std::stringcase`, `std::stringcase`, `epics_list_processing` *, `bool` >
Tuple for filnemae, rule and list processing.
- using `tc_listing_def` = std::vector<`filename_rule_list_tuple` >
List of tuples for filnemae, rule and list processing.
- using `dirname_arg_macro_tuple` = std::tuple<`std::stringcase`, `std::stringcase`, `epics_macrosfiles_processing` *, `const char` * >
Tuple for directory name, argument and macro list processing.
- using `tc_macro_def` = std::vector<`dirname_arg_macro_tuple` >
List of tuples for directory name, argument and macro list processing.

Enumerations

- enum `epics_record_enum` {
 `epics_record_enum::aaival` = 0, `epics_record_enum::aaoval`, `epics_record_enum::aival`, `epics_record_enum::aoval`,
 `epics_record_enum::bival`, `epics_record_enum::boval`, `epics_record_enum::eventval`, `epics_record_enum::histogramval`,
 `epics_record_enum::longinval`, `epics_record_enum::longoutval`, `epics_record_enum::mbbival`, `epics_record_enum::mbboval`,
 `epics_record_enum::mbbiDirectval`, `epics_record_enum::mbboDirectval`, `epics_record_enum::stringinval`,
 `epics_record_enum::stringoutval`,
 `epics_record_enum::lsival`, `epics_record_enum::lsoval`, `epics_record_enum::waveformval`, `epics_record_enum::airval`,
 `epics_record_enum::aoval`, `epics_record_enum::birval`, `epics_record_enum::borval`, `epics_record_enum::mbbiDirectrv`,
 `epics_record_enum::mbboDirectrv`, `epics_record_enum::mbbirval`, `epics_record_enum::mbborval`,
 `epics_record_enum::epics_record_enumEnd`,
 `epics_record_enum::invalidval` = -1 }

Epics record type enum.

Functions

- void `complete_io_scan` (`EpicsInterface` *`epics`, `IOSCANPVT` `ioscan`, int `prio`) noexcept
- int `get_callback_queue_size` (int `pri`)
- int `get_callback_queue_used` (int `pri`)
- int `get_callback_queue_free` (int `pri`)
- int `get_callback_queue_highwatermark` (int `pri`)
- int `get_callback_queue_overflow` (int `pri`)
- int `set_callback_queue_highwatermark_reset` (void)
- const std::regex `tc_regex` ("((tc)::((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +:(8[0-9][0-9]))/(\\d{1,9})/(\\d{1,9}) :(\\\d{1,9}))")
Callback for output record.
- const std::regex `info_regex` ("((tc)::((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +:(8[0-9][0-9]))/(info)/([A-Za-z0-9_+])")
Regex for indentifying info records.
- void `tcLoadRecords` (const iocshArgBuf *`args`)
Load TwinCAT records.
- void `tcSetScanRate` (const iocshArgBuf *`args`) noexcept
Set the scan rate.
- void `tcList` (const iocshArgBuf *`args`)
Generate channel lists.
- void `tcMacro` (const iocshArgBuf *`args`)
Generated macro files.
- void `tcAlias` (const iocshArgBuf *`args`)
Define alias and replacement rules.
- void `tclInfoPrefix` (const iocshArgBuf *`args`)
Sets the info prefix.
- void `tcPrintVals` (const iocshArgBuf *`args`)
Print all values.
- void `tcPrintVal` (const iocshArgBuf *`args`)
Print value.

Variables

- callbackQueueStats `callback_status` {}

7.1.1 Detailed Description

Namespace for TCat device support.

[DevTc](#) Name space

[DevTc](#) namespace

7.1.2 Function Documentation

7.1.2.1 tc_regex()

```
const std::regex DevTc::tc_regex (
    "((tc)://((\\b([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +(8[0-9][0-9])) / (\\d{1,9}) / (\\d{1,9})" ) [inline]
```

Callback for output record.

Callback function to process EPICS out record Regex for indentifying TwinCAT records

Referenced by [DevTc::register_devsup::register_devsup\(\)](#).

Here is the caller graph for this function:

7.1.2.2 tcAlias()

```
void DevTc::tcAlias (
    const iocshArgBuf * args )
```

Define alias and replacement rules.

Define a nick name or alias

Parameters

<i>args</i>	Arguments for tcAlias
-------------	-----------------------

Definition at line 844 of file [drvTc.cpp](#).

References [plc::System::get\(\)](#).

Here is the call graph for this function:

7.1.2.3 tcInfoPrefix()

```
void DevTc::tcInfoPrefix (
    const iocshArgBuf * args )
```

Sets the info prefix.

Sets the channel prefix for info PLC records

Parameters

<i>args</i>	Arguments for tcInfoPrefix
-------------	----------------------------

Definition at line 878 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:

7.1.2.4 tcList()

```
void DevTc::tcList (
    const iocshArgBuf * args )
```

Generate channel lists.

List function to generate separate listings

Parameters

<i>args</i>	Arguments for tcList
-------------	----------------------

Definition at line 794 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:

7.1.2.5 tcLoadRecords()

```
void DevTc::tcLoadRecords (
    const iocshArgBuf * args )
```

Load TwinCAT records.

Function for loading a TCat tpy file, and using it to generate internal record entries as well as the EPICs .db file

Parameters

<i>args</i>	Arguments for tcLoadRecords
-------------	-----------------------------

Definition at line 553 of file drvTc.cpp.

7.1.2.6 tcMacro()

```
void DevTc::tcMacro (
    const iocshArgBuf * args )
```

Generated macro files.

Macro function to generate macro files

Parameters

<i>args</i>	Arguments for tcMacro
-------------	-----------------------

Definition at line 819 of file drvTc.cpp.

References `plc::System::get()`.

Here is the call graph for this function:

7.1.2.7 `tcPrintVal()`

```
void DevTc::tcPrintVal (
    const iocshArgBuf * args )
```

Print value.

Debugging function that prints the values for one or multiple records of the PLCs. Supports wildcards.

Parameters

<i>args</i>	Arguments for tcPrintVal
-------------	--------------------------

Definition at line 920 of file drvTc.cpp.

References `plc::System::get()`, and `plc::System::printVal()`.

Here is the call graph for this function:

7.1.2.8 `tcPrintVals()`

```
void DevTc::tcPrintVals (
    const iocshArgBuf * args )
```

Print all values.

Debugging function that prints the values for all records of the PLCs

Parameters

<i>args</i>	Arguments for tcPrintVals
-------------	---------------------------

Definition at line 909 of file drvTc.cpp.

References `plc::System::get()`, and `plc::System::printVals()`.

Here is the call graph for this function:

7.1.2.9 `tcSetScanRate()`

```
void DevTc::tcSetScanRate (
    const iocshArgBuf * args ) [noexcept]
```

Set the scan rate.

Set scan rate of the read scanner

Parameters

<code>args</code>	Arguments for <code>tcSetScanRate</code>
-------------------	--

Definition at line 737 of file `drvTc.cpp`.

References `plc::System::get()`.

Here is the call graph for this function:

7.2 EpicsTpy Namespace Reference

Namespace for tpy-db conversion.

Classes

- class [epics_conversion](#)
Epics conversion.
- class [epics_db_processing](#)
pics database record processing
- class [epics_list_processing](#)
List processing.
- class [epics_macrofiles_processing](#)
Macro file processing.
- struct [macro_info](#)
Macro information.
- struct [macro_record](#)
Macro record.
- class [multi_io_support](#)
Multiple IO support.
- class [split_io_support](#)
Split IO support.

Typedefs

- using `macro_list` = `std::vector<macro_info>`
- using `macro_stack` = `std::stack<macro_record>`
- using `filename_set` = `std::unordered_set<std::stringcase>`

Enumerations

- enum `tc_epics_conv` { `tc_epics_conv::no_conversion`, `tc_epics_conv::no_dot`, `tc_epics_conv::ligo_std`, `tc_epics_conv::ligo_vac` }

Conversion rules for TC/EPICS.
- enum `case_type` { `case_type::preserve`, `case_type::upper`, `case_type::lower` }

Case conversion rule enum.
- enum `io_filestat` { `io_filestat::closed`, `read`, `write` }

enum for file io
- enum `listing_type` { `listing_type::standard`, `listing_type::autoburt`, `listing_type::daqini` }

Listing type enum.
- enum `macrofile_type` { `macrofile_type::all`, `macrofile_type::fields`, `macrofile_type::errors` }
- enum `device_support_type` { `device_support_type::opc_name`, `device_support_type::tc_name` }

Device support enum.
- enum `string_support_type` { `string_support_type::short_string`, `string_support_type::long_string`, `vary_string` }

String support enum.
- enum `int_support_type` { `int_support_type::int_32`, `int_support_type::int_64`, `int_auto` }

Integer support enum.

Variables

- constexpr int `MAX_EPICS_CHANNEL` = 54
- constexpr int `MAX_EPICS_DESC` = 40
- constexpr int `MAX_EPICS_STRING` = 40
- constexpr int `MAX_EPICS_LONGSTRING` = 65535
- constexpr int `MAX_EPICS_UNIT` = 15
- constexpr int `MAX_EPICS_ENUM` = 25
- const char *const `EPICS_DB_EGU` = "EGU"
- const char *const `EPICS_DB_DESC` = "DESC"
- const char *const `EPICS_DB_HOPR` = "HOPR"
- const char *const `EPICS_DB_LOPR` = "LOPR"
- const char *const `EPICS_DB_DRVH` = "DRVH"
- const char *const `EPICS_DB_DRVL` = "DRVL"
- const char *const `EPICS_DB_ONAM` = "ONAM"
- const char *const `EPICS_DB_ZNAM` = "ZNAM"
- const char *const `EPICS_DB_PREC` = "PREC"
- const char *const `EPICS_DB_ZRST` [16]
- const char *const `EPICS_DB_ZRVL` [16]
- const char *const `EPICS_DB_SCAN` = "SCAN"
- const char *const `EPICS_DB_INP` = "INP"
- const char *const `EPICS_DB_OUT` = "OUT"
- const char *const `EPICS_DB_TSE` = "TSE"
- const char *const `EPICS_DB_PINI` = "PINI"
- const char *const `EPICS_DB_DTYP` = "DTYP"
- const char *const `EPICS_DB_SIZV` = "SIZV"
- const char *const `EPICS_DB_OSV` = "OSV"
- const char *const `EPICS_DB_ZSV` = "ZSV"
- const char *const `EPICS_DB_COSV` = "COSV"
- const char *const `EPICS_DB_HIHI` = "HIHI"
- const char *const `EPICS_DB_HIGH` = "HIGH"
- const char *const `EPICS_DB_LOW` = "LOW"
- const char *const `EPICS_DB_LOLO` = "LOLO"

- const char *const EPICS_DB_HYST = "HYST"
- const char *const EPICS_DB_HHSV = "HHSV"
- const char *const EPICS_DB_HSV = "HSV"
- const char *const EPICS_DB_LSV = "LSV"
- const char *const EPICS_DB_LLSV = "LLSV"
- const char *const EPICS_DB_NOALARM = "NO_ALARM"
- const char *const EPICS_DB_MINOR = "MINOR"
- const char *const EPICS_DB_MAJOR = "MAJOR"
- const char *const EPICS_DB_UNSV = "UNSV"
- const char *const EPICS_DB_ZRSV [16]
- const char *const EPICS_DB_FORBIDDEN []
- const char *const EPICS_DB_ALLOWED []
- const char *const EPICS_DB_NUMVAL []
- const char *const LIGODAQ_DATATYPE_NAME = "datatype"
- constexpr int LIGODAQ_DATATYPE_FLOAT = 4
- constexpr int LIGODAQ_DATATYPE_INT32 = 2
- constexpr int LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
- const char *const LIGODAQ_UNIT_NAME = "units"
- const char *const LIGODAQ_UNIT_NONE = "none"
- const char *const LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
- const char *const LIGODAQ_INI_HEADER

7.2.1 Detailed Description

Namespace for tpy-db conversion.

[EpicsTpy](#) name space

7.3 InfoPlc Namespace Reference

Namespace for Info communication.

Classes

- class [InfoInterface](#)
Info interface.
- class [process_arg_info](#)
Arguments for processing.

Typedefs

- using [info_update_method](#) = bool(InfoInterface::*)()
Pointer to info update method.
- using [info_dbrecord_type](#) = std::tuple< ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method >
- using [info_dbrecord_list](#) = std::vector<info_dbrecord_type>
List type of db info tuples.

Enumerations

- enum `update_enum` { `update_enum::forever`, `update_enum::once`, `update_enum::done` }
Update frequency type.

Variables

- constexpr `time_t WINDOWS_TICK` = 10'000'000
- constexpr `time_t SEC_TO_UNIX_EPOCH` = 11'644'473'600LL

7.3.1 Detailed Description

Namespace for Info communication.

`InfoPlc` name space, which has all the classes and functions used for communicating with Info.

7.4 ParseTpy Namespace Reference

Namespace for parsing.

Classes

- class `ads_routing_info`
ADS routing information.
- class `base_record`
Base record definition.
- class `compiler_info`
Compiler information.
- class `item_record`
item record
- class `parserinfo_type`
Parser information.
- class `project_record`
Project information.
- class `symbol_record`
Symbol record.
- class `tpy_file`
Tpy file parsing.
- class `type_map`
Type dictionary.
- class `type_record`
Type record information.

Typedefs

- using `dimension` = std::pair< int, int >
- using `dimensions` = std::list< `dimension` >
- using `enum_map` = std::map< int, std::stringcase >
- using `enum_pair` = std::pair< int, std::stringcase >
- using `item_list` = std::list< `item_record` >
- using `type_multimap` = std::multimap< unsigned int, `type_record` >
- using `symbol_list` = std::list< `symbol_record` >

Enumerations

- enum `type_enum` {
 `type_enum::unknown`, `type_enum::simple`, `type_enum::arraytype`, `type_enum::enumtype`,
 `type_enum::structtype`, `type_enum::functionblock` }

Type enum.

Functions

- bool `compareNamesWoNamespace` (const std::stringcase &p1, const std::stringcase &p2)
- bool `get_decoration` (const char **atts, unsigned int &decoration)
- bool `get_pointer` (const char **atts)

Variables

- const char *const `xmlPlcProjectInfo` = "PlcProjectInfo"
PLC project info.
- const char *const `xmlProjectInfo` = "ProjectInfo"
Project info.
- const char *const `xmlRoutingInfo` = "RoutingInfo"
Routing info.
- const char *const `xmlCompilerInfo` = "CompilerInfo"
Compiler info.
- const char *const `xmlAdsInfo` = "AdsInfo"
ADS info.
- const char *const `xmlDataTypes` = "DataTypes"
Data types.
- const char *const `xmlDataType` = "DataType"
Data type.
- const char *const `xmlSymbols` = "Symbols"
Symbols.
- const char *const `xmlSymbol` = "Symbol"
Symbol.
- const char *const `xmlProperties` = "Properties"
Properties.
- const char *const `xmlProperty` = "Property"
Property.
- const char *const `xmlCompilerVersion` = "CompilerVersion"
Compiler version.
- const char *const `xmlTwinCATVersion` = "TwinCATVersion"

- TwinCAT version.*
- const char *const **xmlCpuFamily** = "CpuFamily"
CPU family.
 - const char *const **xmlNetId** = "NetId"
Net ID.
 - const char *const **xmlPort** = "Port"
Port.
 - const char *const **xmlTargetName** = "TargetName"
Target name.
 - const char *const **xmlName** = "Name"
Name.
 - const char *const **xmlType** = "Type"
Type.
 - const char *const **xmlAttrDecoration** = "Decoration"
Decoration.
 - const char *const **xmlAttrPointer** = "Pointer"
Pointer.
 - const char *const **xmlIGroup** = "IGroup"
I Group.
 - const char *const **xmlIOffset** = "IOffset"
I Offset.
 - const char *const **xmlBitSize** = "BitSize"
Bit size.
 - const char *const **xmlBitOffs** = "BitOffs"
Bit Offset.
 - const char *const **xmlArrayInfo** = "ArrayInfo"
Array info.
 - const char *const **xmlArrayLBound** = "LBound"
Lower bound.
 - const char *const **xmlArrayElements** = "Elements"
Elements.
 - const char *const **xmlSubItem** = "SubItem"
Sub item.
 - const char *const **xmlFbInfo** = "FbInfo"
Fb info.
 - const char *const **xmlEnumInfo** = "EnumInfo"
Enum info.
 - const char *const **xmlEnumText** = "Text"
Text.
 - const char *const **xmlEnumEnum** = "Enum"
Enum.
 - const char *const **xmlEnumComment** = "Comment"
Comment.
 - const char *const **xmlValue** = "Value"
Value.
 - const char *const **xmlDesc** = "Desc"
Description.
 - const char *const **opcExport** = "opc"
OPC.
 - const char *const **opcProp** = "opc_prop"
OPC property.

- const char *const [opcBracket](#) = "["
OPC bracket.
- const char *const [xmlSubstitution](#) = "TcSubstitution"
Substitution.
- const char *const [xmlAlias](#) = "Alias"
Substitution EPICS channel name.

7.4.1 Detailed Description

Namespace for parsing.

Namespace for tpy parsing.

[ParseTpy](#) name space

7.4.2 Function Documentation

7.4.2.1 compareNamesWoNamespace()

```
bool ParseTpy::compareNamesWoNamespace (
    const std::stringcase & p1,
    const std::stringcase & p2 )
```

compareNamesWoNamespace

Definition at line 349 of file ParseTpy.cpp.

7.4.2.2 get_decoration()

```
bool ParseTpy::get_decoration (
    const char ** attrs,
    unsigned int & decoration )
```

XML get decoration number from attribute

Definition at line 535 of file ParseTpy.cpp.

References [xmlAttrDecoration](#).

7.4.2.3 get_pointer()

```
bool ParseTpy::get_pointer (
    const char ** attrs )
```

XML get pointer from attribute

Definition at line 549 of file ParseTpy.cpp.

References `xmlAttrPointer`.

7.5 ParseUtil Namespace Reference

Namespace for parsing utilities.

Classes

- class [bit_location](#)
Bit location.
- class [memory_location](#)
Memory location.
- class [opc_list](#)
OPC list.
- class [optarg](#)
Optional arguments.
- class [parserinfo_type](#)
Parser information.
- class [process_arg](#)
Arguments for processing.
- class [process_arg_tc](#)
Arguments for processing.
- class [replacement_rules](#)
Replacement rules.
- class [substitution](#)
Channel substitution.
- class [substitution_list](#)
List of channel substitutions.
- class [tag_processing](#)
Tag processing selection.
- class [variable_name](#)
Variable name.

Typedefs

- using [replacement_table](#) = std::map< `std::stringcase`, `std::stringcase` >
- using [property_map](#) = std::map< `int`, `std::stringcase` >
- using [property_el](#) = std::pair< `int`, `std::stringcase` >
- using [substitution_map](#) = std::map< `std::stringcase`, [substitution](#) >
Map of channel substitution.

Enumerations

- enum `opc_enum` { `opc_enum::no_change`, `opc_enum::publish`, `opc_enum::silent` }
OPC state enum.
- enum `process_type_enum` {
 `process_type_enum::pt_invalid`, `process_type_enum::pt_int`, `process_type_enum::pt_real`, `process_type_enum::pt_bool`,
 `process_type_enum::pt_string`, `process_type_enum::pt_enum`, `process_type_enum::pt_binary` }
Process type.
- enum `process_tag_enum` { `process_tag_enum::all`, `process_tag_enum::atomic`, `process_tag_enum::structured` }
Tag processing enum.
- enum `process_substitution_enum` { `process_substitution_enum::standard`, `process_substitution_enum::all`,
 `process_substitution_enum::ignore` }
Substitution processing enum.

Variables

- `constexpr int OPC_PROP_CDT = 1`
- `constexpr int OPC_PROP_VALUE = 2`
- `constexpr int OPC_PROP_QUALITY = 3`
- `constexpr int OPC_PROP_TIME = 4`
- `constexpr int OPC_PROP_RIGHTS = 5`
- `constexpr int OPC_PROP_SCANRATE = 6`
- `constexpr int OPC_PROP_UNIT = 100`
- `constexpr int OPC_PROP_DESC = 101`
- `constexpr int OPC_PROP_HIEU = 102`
- `constexpr int OPC_PROP_LOEU = 103`
- `constexpr int OPC_PROP_HIRANGE = 104`
- `constexpr int OPC_PROP_LORANGE = 105`
- `constexpr int OPC_PROP_CLOSE = 106`
- `constexpr int OPC_PROP_OPEN = 107`
- `constexpr int OPC_PROP_TIMEZONE = 108`
- `constexpr int OPC_PROP_FGC = 201`
- `constexpr int OPC_PROP_BGC = 202`
- `constexpr int OPC_PROP_BLINK = 203`
- `constexpr int OPC_PROP_BMP = 204`
- `constexpr int OPC_PROP SND = 205`
- `constexpr int OPC_PROP_HTML = 206`
- `constexpr int OPC_PROP_AVI = 207`
- `constexpr int OPC_PROP_ALMSTAT = 300`
- `constexpr int OPC_PROP_ALMHELP = 301`
- `constexpr int OPC_PROP_ALMAREAS = 302`
- `constexpr int OPC_PROP_ALMPRIMARYAREA = 303`
- `constexpr int OPC_PROP_ALMCONDITION = 304`
- `constexpr int OPC_PROP_ALMLIMIT = 305`
- `constexpr int OPC_PROP_ALMDB = 306`
- `constexpr int OPC_PROP_ALMH = 307`
- `constexpr int OPC_PROP_ALMH = 308`
- `constexpr int OPC_PROP_ALML = 309`
- `constexpr int OPC_PROP_ALMLL = 310`
- `constexpr int OPC_PROP_ALMROC = 311`
- `constexpr int OPC_PROP_ALMDEV = 312`
- `constexpr int OPC_PROP_PREC = 8500`

- `constexpr int OPC_PROP_ZRST = 8510`
- `constexpr int OPC_PROP_FFST = 8525`
- `constexpr int OPC_PROP_RECTYPE = 8600`
- `constexpr int OPC_PROP_INOUT = 8601`
- `const char *const OPC_PROP_INPUT = "input"`
- `const char *const OPC_PROP_OUTPUT = "output"`
- `constexpr int OPC_PROP_TSE = 8602`
- `constexpr int OPC_PROP_PINI = 8603`
- `constexpr int OPC_PROP_DTYP = 8604`
- `constexpr int OPC_PROP_SERVER = 8610`
- `constexpr int OPC_PROP_PLNAME = 8611`
- `constexpr int OPC_PROP_ALIAS = 8620`
- `const char *const OPC_NAME_ALIAS = "ALIAS"`
- `constexpr int OPC_PROP_ALMOSV = 8700`
- `constexpr int OPC_PROP_ALMZSV = 8701`
- `constexpr int OPC_PROP_ALMCOSV = 8702`
- `constexpr int OPC_PROP_ALMUNSV = 8703`
- `constexpr int OPC_PROP_ALMZRSV = 8710`
- `constexpr int OPC_PROP_ALMFFSV = 8725`
- `constexpr int OPC_PROP_ALMHHSV = 8727`
- `constexpr int OPC_PROP_ALMHHSV = 8728`
- `constexpr int OPC_PROP_ALMLSV = 8729`
- `constexpr int OPC_PROP_ALMLLSV = 8730`
- `constexpr int OPC_PROP_FIELD_BEG = 8800`
- `constexpr int OPC_PROP_FIELD_END = 9000`

7.5.1 Detailed Description

Namespace for parsing utilities.

[ParseUtil](#) name space

[ParseTpy](#) name space

7.6 plc Namespace Reference

Namespace for abstract plc functionality.

Classes

- class [BasePLC](#)
Base PLC.
- class [BaseRecord](#)
Class for managing a tag/channel.
- class [DataValue](#)
Data value.
- struct [DataValueTraits](#)
Data value traits.
- struct [DataValueTypeDef](#)
Collection of type definitions.
- class [Interface](#)
Abstract interface.
- class [System](#)
System to keep track of PLCs.

Typedefs

- using `scanner_thread_args` = struct { `plc::BasePLC *plc`
`Scanner thread arguments.`
- using `BasePLCPtr` = `std::shared_ptr< BasePLC >`
`Smart pointer to PLC.`
- using `InterfacePtr` = `std::unique_ptr< Interface >`
`Smart pointer to interface.`
- using `BaseRecordPtr` = `std::shared_ptr< BaseRecord >`
`smart pointer to record`
- using `BaseRecordList` = `std::unordered_map< std::stringcase, BaseRecordPtr >`
`list of record`
- using `BasePLCList` = `std::map< std::stringcase, BasePLCPtr >`
`BasePLC map.`

Enumerations

- enum `data_type_enum` {
`data_type_enum::dtInvalid, data_type_enum::dtBool, data_type_enum::dtInt8, data_type_enum::dtUInt8,`
`data_type_enum::dtInt16, data_type_enum::dtUInt16, data_type_enum::dtInt32, data_type_enum::dtUInt32,`
`data_type_enum::dtInt64, data_type_enum::dtUInt64, data_type_enum::dtFloat, data_type_enum::dtDouble,`
`data_type_enum::dtString, data_type_enum::dtWString, data_type_enum::dtBinary }`
`Data type enumeration.`
- enum `access_rights_enum` { `access_rights_enum::read_only, access_rights_enum::write_only, access_rights_enum::read_write`
`}`
`Access rights enum.`

Functions

- template<> `bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, const DataValueTypeDef::atomic_string *source) noexcept`
`Reset and read.`
- template<> `bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source) noexcept`
`Write and test.`
- VOID CALLBACK `ScannerProc (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)`
`Scanner thread callback.`
- `DWORD WINAPI scannerThread (scanner_thread_args args) noexcept`
`Scanner thread.`
- template<typename T , typename U >
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, U source) noexcept`
`Reset and read.`
- template<typename T >
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, const typename DataValueTraits< T >::traits_atomic *source) noexcept`
`Reset and read.`
- template<typename T , typename U >
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, U dest, const T &source) noexcept`
`Write and test.`
- template<typename T >
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest, const T &source) noexcept`
`Write and test.`

Variables

- long `scanperiod`
Period in ms of the scanner.
- `plc::BasePLC::scanner_func scanner`
Address of the scanner function to be used (read, write, or upddate)

7.6.1 Detailed Description

Namespace for abstract plc functionality.

PLC namespace

7.6.2 Typedef Documentation

7.6.2.1 BasePLCList

```
using plc::BasePLCList = typedef std::map<std::stringcase, BasePLCPtr>
```

BasePLC map.

This is list of `BasePLC`, ordered by their name.

Definition at line 893 of file plcBase.h.

7.6.2.2 BasePLCPtr

```
using plc::BasePLCPtr = typedef std::shared_ptr<BasePLC>
```

Smart pointer to PLC.

This is a smart pointer to a PLC

Definition at line 26 of file plcBase.h.

7.6.2.3 BaseRecordList

```
using plc::BaseRecordList = typedef std::unordered_map<std::stringcase, BaseRecordPtr>
```

list of record

This is a list of tag/channel records organized as a hash map

Definition at line 711 of file plcBase.h.

7.6.2.4 BaseRecordPtr

```
using plc::BaseRecordPtr = typedef std::shared_ptr<BaseRecord>
```

smart pointer to record

This is a smart pointer to a tag/channel record

Definition at line 706 of file plcBase.h.

7.6.2.5 InterfacePtr

```
using plc::InterfacePtr = typedef std::unique_ptr<Interface>
```

Smart pointer to interface.

This is a smart pointer for [Interface](#)

Definition at line 69 of file plcBase.h.

7.6.2.6 scanner_thread_args

```
using plc::scanner_thread_args = typedef struct { plc::BasePLC* plc
```

Scanner thread arguments.

Structure for arguments sent to a scanner thread

Definition at line 644 of file plcBase.cpp.

7.6.3 Enumeration Type Documentation

7.6.3.1 access_rights_enum

```
enum plc::access_rights_enum [strong]
```

Access rights enum.

Enum for access rights of a record

Enumerator

read_only	Read only.
write_only	Write only.
read_write	Read/write.

Definition at line 480 of file plcBase.h.

7.6.3.2 data_type_enum

```
enum plc::data\_type\_enum [strong]
```

Data type enumeration.

This is an enumerated type listing all the available data types

Enumerator

<code>dtInvalid</code>	Invalid data type.
<code>dtBool</code>	Boolean.
<code>dtInt8</code>	1-byte integer
<code>dtUInt8</code>	1-byte unsigned integer
<code>dtInt16</code>	2-byte integer
<code>dtUInt16</code>	2-byte unsigned integer
<code>dtInt32</code>	4-byte integer
<code>dtUInt32</code>	4-byte unsigned integer
<code>dtInt64</code>	8-byte integer
<code>dtUInt64</code>	8-byte unsigned integer
<code>dtFloat</code>	4-byte single precision floating point
<code>dtDouble</code>	8-byte double precision floating point
<code>dtString</code>	string class
<code>dtWString</code>	wstring class
<code>dtBinary</code>	binary object

Definition at line 74 of file plcBase.h.

7.6.4 Function Documentation

7.6.4.1 reset_and_read() [1/3]

```
template<>
bool plc::reset\_and\_read (
    DataValueTypeDef::atomic\_bool & dirty,
    DataValueTypeDef::type\_wstring & dest,
    const DataValueTypeDef::atomic\_string * source) [noexcept]
```

Reset and read.

Will read the value and reset the dirty flag (wstring from string).

Definition at line 31 of file plcBase.cpp.

References `plc::DataValueTypeDef::memory_order`.

Referenced by `plc::DataValue::Read()`.

Here is the caller graph for this function:

7.6.4.2 `reset_and_read()` [2/3]

```
template<typename T >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    const typename DataValueTraits< T >::traits_atomic * source ) [noexcept]
```

Reset and read.

Will read the value and reset the dirt flag (same type).

Definition at line 33 of file plcBaseTemplate.h.

References `plc::DataValueTypeDef::memory_order`.

7.6.4.3 `reset_and_read()` [3/3]

```
template<typename T , typename U >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    U source ) [noexcept]
```

Reset and read.

Will read the value and reset the dirty flag.

Definition at line 15 of file plcBaseTemplate.h.

References `plc::DataValueTypeDef::memory_order`.

7.6.4.4 `ScannerProc()`

```
VOID CALLBACK plc::ScannerProc (
    LPVOID lpArg,
    DWORD dwTimerLowValue,
    DWORD dwTimerHighValue )
```

Scanner thread callback.

Scanner thread callback with periodic timer

Definition at line 657 of file plcBase.cpp.

Referenced by `scannerThread()`.

Here is the caller graph for this function:

7.6.4.5 scannerThread()

```
DWORD WINAPI plc::scannerThread (
    scanner_thread_args args ) [noexcept]
```

Scanner thread.

Scanner thread with periodic timer This function uses the windows waitable timer which will call a completion routine at a regular interval. The completion routine in this case is one of either read_scanner, write_scanner, or update_scanner.

Definition at line 676 of file plcBase.cpp.

References ScannerProc().

Referenced by plc::BasePLC::start_read_scanner(), plc::BasePLC::start_update_scanner(), and plc::BasePLC::start_write_scanner().

Here is the call graph for this function: Here is the caller graph for this function:

7.6.4.6 write_and_test() [1/3]

```
template<>
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    DataValueTypeDef::atomic_wstring * dest,
    const DataValueTypeDef::type_string & source ) [noexcept]
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (string to wstring).

Definition at line 53 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order.

Referenced by plc::DataValue::Write().

Here is the caller graph for this function:

7.6.4.7 write_and_test() [2/3]

```
template<typename T >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    typename DataValueTraits< T >::traits_atomic * dest,
    const T & source ) [noexcept]
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (same type).

Definition at line 77 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.6.4.8 `write_and_test()` [3/3]

```
template<typename T , typename U >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    U dest,
    const T & source ) [noexcept]
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one.

Definition at line 52 of file plcBaseTemplate.h.

References `plc::DataValueTypeDef::memory_order`.

7.7 TcComms Namespace Reference

Namespace for TCat communication.

Classes

- class [AmsRouterNotification](#)
AMS Router Notification.
- struct [DataPar](#)
Memory location struct.
- class [TCatInterface](#)
TCat interface class.
- class [TcPLC](#)
TwinCAT PLC.
- class [tcProcWrite](#)
TwinCAT process write requests.

Functions

- void [errorPrintf](#) (int nErr) noexcept
errorPrintf
- void __stdcall [ADSCallback](#) (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plclId)
- void __stdcall [RouterCall](#) (long nReason)

Variables

- `constexpr time_t WINDOWS_TICK = 10'000'000`
- `constexpr time_t SEC_TO_UNIX_EPOCH = 11'644'473'600LL`
- `constexpr int MAX_REQ_SIZE = 250000`
 $\text{maximum allowed request size (bytes)}$
- `constexpr int MAX_SINGLE_GAP_SIZE = 50`
 $\text{maximum allowed size (bytes) of a memory gap within continuous request}$
- `constexpr double MAX_REL_GAP = 0.25`
 $(\text{maximum allowed total gap size}) / (\text{current request size})$
- `constexpr int MIN_REL_GAP_SIZE = 100`
 $\text{minimum allowed relative gap size (bytes)}$
- `constexpr int default_scanrate = 100`
 $\text{default PLC TwinCAT scan rate (100ms)}$
- `constexpr int minimum_scanrate = 5`
 $\text{minimum PLC TwinCAT scan rate (5ms)}$
- `constexpr int maximum_scanrate = 10000`
 $\text{maximum PLC TwinCAT scan rate (10s)}$
- `constexpr int default_multiple = 10`
 $\text{default multiple for PLC EPICS scan rate (10)}$
- `constexpr int minimum_multiple = 1`
 $\text{minimum multiple for PLC EPICS scan rate (1)}$
- `constexpr int maximum_multiple = 200`
 $\text{maximum multiple for PLC EPICS scan rate (200)}$

7.7.1 Detailed Description

Namespace for TCat communication.

`TcComms` name space, which has all the classes and functions used for communicating with TCat.

7.7.2 Function Documentation

7.7.2.1 `ADScallback()`

```
void __stdcall TcComms::ADScallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId )
```

Callback for ADS state change

Definition at line 683 of file tcComms.cpp.

References `TcComms::TcPLC::set_ads_state()`.

Here is the call graph for this function:

7.7.2.2 errorPrintf()

```
void TcComms::errorPrintf (
    int nErr ) [noexcept]
```

errorPrintf

Print an error message for an ADS error return code

Definition at line 31 of file tcComms.cpp.

Referenced by TcComms::TcPLC::read_scanner(), TcComms::TcPLC::remove_ads_notification(), TcComms::TcPLC::setup_ads_notification(), and TcComms::tcProcWrite::tcwrite().

Here is the caller graph for this function:

7.7.2.3 RouterCall()

```
void __stdcall TcComms::RouterCall (
    long nReason )
```

Callback for AMS router state change

Definition at line 913 of file tcComms.cpp.

Chapter 8

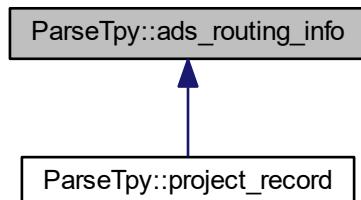
Class Documentation

8.1 ParseTpy::ads_routing_info Class Reference

ADS routing information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::ads_routing_info:



Public Member Functions

- `ads_routing_info () noexcept=default`
Default constructor.
- `ads_routing_info (const std::string& netid, int port=801) noexcept`
Constructor.
- `ads_routing_info (const std::string& netid, int port, const std::string& targetname) noexcept`
Constructor.
- `const std::string& get_netid () const noexcept`
Get ADS net id.
- `void set_netid (const std::string& netid) noexcept`
Set ADS net id.
- `int get_port () const noexcept`

- `void set_port (int port) noexcept`
Set ADS port.
- `const std::stringcase & get_targetname () const noexcept`
Get ADS target name.
- `void set_targetname (const std::stringcase &targetname)`
Set ADS target name.
- `bool isValid () const noexcept`
Checks, if net id is of the form n.n.n.n.n.
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`
- `bool get (unsigned char &a1, unsigned char &a2, unsigned char &a3, unsigned char &a4, unsigned char &a5, unsigned char &a6) const noexcept`

Protected Attributes

- `std::stringcase ads_netid`
ADS net ID.
- `int ads_port = 0`
ADS port.
- `std::stringcase ads_targetname`
ADS target name.

8.1.1 Detailed Description

ADS routing information.

This is a base class for storing the ADS routing information

Definition at line 23 of file ParseTpy.h.

8.1.2 Member Function Documentation

8.1.2.1 `get()` [1/2]

```
std::stringcase ParseTpy::ads_routing_info::get ( ) const
```

Gets a string representation of a ads routing information

Returns

string with format "tc://netid:port/", empty on error

Definition at line 215 of file ParseTpy.cpp.

8.1.2.2 get() [2/2]

```
bool ParseTpy::ads_routing_info::get (
    unsigned char & a1,
    unsigned char & a2,
    unsigned char & a3,
    unsigned char & a4,
    unsigned char & a5,
    unsigned char & a6 ) const [noexcept]
```

Get address in net format

Parameters

<i>a1</i>	First address qualifier (return)
<i>a2</i>	Second address qualifier (return)
<i>a3</i>	Third address qualifier (return)
<i>a4</i>	Fourth address qualifier (return)
<i>a5</i>	Fifth address qualifier (return)
<i>a6</i>	Sixth address qualifier (return)

Definition at line 228 of file ParseTpy.cpp.

8.1.2.3 set()

```
bool ParseTpy::ads_routing_info::set (
    const std::string& s )
```

Set the ads routing information using a string of the form: "tc://netid:port/" where netid is a string of the format n.n.n.n.n

Parameters

<i>s</i>	String describing the ads routing information
----------	---

Returns

True if successful

Definition at line 249 of file ParseTpy.cpp.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.2 TcComms::AmsRouterNotification Class Reference

AMS Router Notification.

```
#include <tcComms.h>
```

Public Member Functions

- int [get_ads_version \(\) const noexcept](#)
get ADS protocol/library version
- int [get_ads_revision \(\) const noexcept](#)
get ADS protocol/library revision
- int [get_ads_build \(\) const noexcept](#)
get ADS protocol/library build

Static Public Member Functions

- static AmsRouterEvent [get_router_notification \(\) noexcept](#)
get router notification
- static const [AmsRouterNotification & get_instance \(\) noexcept](#)
get global instance

Protected Attributes

- int [ads_version](#)
ADS protocol/library version.
- int [ads_revision](#)
ADS protocol/library revision.
- int [ads_build](#)
ADS protocol/library build.

Friends

- void __stdcall [RouterCall \(long\)](#)
Notification callback is a friend.

8.2.1 Detailed Description

AMS Router Notification.

Class for a AMS router notifications

Definition at line 409 of file tcComms.h.

8.2.2 Friends And Related Function Documentation

8.2.2.1 RouterCall

```
void __stdcall RouterCall (
    long nReason ) [friend]
```

Notification callback is a friend.

Callback for AMS router state change

Definition at line 913 of file tcComms.cpp.

The documentation for this class was generated from the following files:

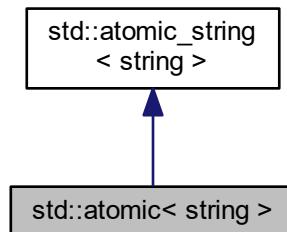
- [tcComms.h](#)
- [tcComms.cpp](#)

8.3 std::atomic< string > Class Reference

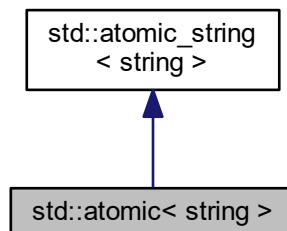
[atomic<string>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< string >:



Collaboration diagram for std::atomic< string >:



Additional Inherited Members

8.3.1 Detailed Description

[atomic<string>](#)

This is a class implements an atomic specialization for std::string

Definition at line 75 of file atomic_string.h.

The documentation for this class was generated from the following file:

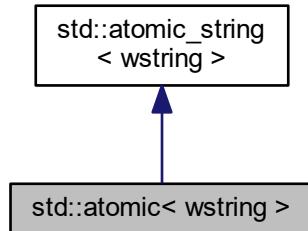
- [atomic_string.h](#)

8.4 std::atomic< wstring > Class Reference

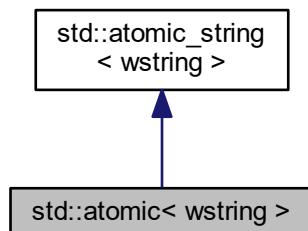
[atomic<wstring>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< wstring >:



Collaboration diagram for std::atomic< wstring >:



Additional Inherited Members

8.4.1 Detailed Description

[atomic<wstring>](#)

This is a class implements an atomic specialization for std::wstring

Definition at line 81 of file atomic_string.h.

The documentation for this class was generated from the following file:

- [atomic_string.h](#)

8.5 std::atomic_string< stringT > Class Template Reference

atomic strings

```
#include <atomic_string.h>
```

Public Member Functions

- [atomic_string \(\) noexcept=default](#)
Default constructor.
- [atomic_string \(const stringT &s\)](#)
Constructor from data.
- [stringT operator= \(const stringT &right\)](#)
Assignment operator on basic type.
- [bool is_lock_free \(\) const](#)
Not lock free.
- [void store \(stringT value, memory_order order=memory_order_seq_cst\)](#)
Store.
- [stringT load \(memory_order order=memory_order_seq_cst\) const](#)
Load.
- [operator stringT \(\) const](#)
Convert to string.
- [stringT exchange \(const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Exchange.
- [bool compare_exchange_weak \(stringT &exp, const stringT &value, memory_order order1, memory_order order2\)](#)
Compare exchange.
- [bool compare_exchange_weak \(stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Compare exchange.
- [bool compare_exchange_strong \(stringT &exp, const stringT &value, memory_order order1, memory_order order2\)](#)
Compare exchange.
- [bool compare_exchange_strong \(stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Compare exchange.

Protected Attributes

- atomic_flag `flag` = ATOMIC_FLAG_INIT
flag for spin lock
- stringT `data`
data string

8.5.1 Detailed Description

```
template<typename stringT>
class std::atomic_string< stringT >
```

atomic strings

This is a class implements an atomic specialization for strings

Definition at line 16 of file atomic_string.h.

8.5.2 Member Function Documentation

8.5.2.1 operator=()

```
template<typename stringT >
stringT std::atomic_string< stringT >::operator= (
    const stringT & right )
```

Assignment operator on basic type.

Assignment operator.

Definition at line 86 of file atomic_string.h.

The documentation for this class was generated from the following file:

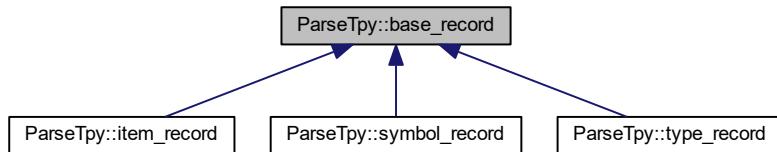
- [atomic_string.h](#)

8.6 ParseTpy::base_record Class Reference

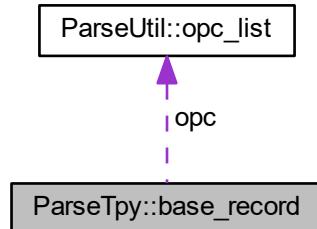
Base record definition.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::base_record:



Collaboration diagram for ParseTpy::base_record:



Public Member Functions

- `base_record () noexcept=default`
Default constructor.
- `base_record (const std::stringcase &n)`
- `base_record (const std::stringcase &n, const ParseUtil::opc_list &o)`
- `base_record (const std::stringcase &n, const ParseUtil::opc_list &o, const std::stringcase &tn, unsigned int td=0)`
- `base_record (const std::stringcase &n, const std::stringcase &tn, unsigned int td=0)`
- `const std::stringcase & get_name () const noexcept`
Get name.
- `std::stringcase & get_name () noexcept`
Get name.
- `void set_name (std::stringcase n)`
Set name.
- `const std::stringcase & get_type_name () const noexcept`

- `std::stringcase & get_type_name () noexcept`
Get type name.
- `void set_type_name (std::stringcase t) noexcept`
Set type name.
- `unsigned int get_type_decoration () const noexcept`
Get type decoration.
- `void set_type_decoration (unsigned int id) noexcept`
Set type decoration.
- `bool get_type_pointer () const noexcept`
Get type pointer.
- `void set_type_pointer (bool isPointer) noexcept`
Set type pointer.
- `const ParseUtil::opc_list & get_opc () const noexcept`
Get OPC list.
- `ParseUtil::opc_list & get_opc () noexcept`
Get OPC list.

Protected Attributes

- `std::stringcase name`
name of type
- `std::stringcase type_n`
type definition
- `unsigned int type_decoration = 0`
decoration or type ID of type definition
- `bool type_pointer = false`
this is a pointer
- `ParseUtil::opc_list opc`
list of opc properties

8.6.1 Detailed Description

Base record definition.

This is a base class for storing name, type, type id and opc list

Definition at line 148 of file ParseTpy.h.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 `base_record()` [1/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n ) [inline], [explicit]
```

Constructor

Parameters

<i>n</i>	Name
----------	------

Definition at line 155 of file ParseTpy.h.

8.6.2.2 base_record() [2/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o ) [inline]
```

Constructor**Parameters**

<i>n</i>	Name
<i>o</i>	OPC list

Definition at line 159 of file ParseTpy.h.

8.6.2.3 base_record() [3/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o,
    const std::stringcase & tn,
    unsigned int td = 0 ) [inline]
```

Constructor**Parameters**

<i>n</i>	Name
<i>o</i>	OPC list
<i>tn</i>	Type name
<i>td</i>	Type decortation or id

Definition at line 166 of file ParseTpy.h.

8.6.2.4 base_record() [4/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
```

```
const std::stringcase & tn,
unsigned int td = 0 ) [inline]
```

Constructor

Parameters

<i>n</i>	Name
<i>tn</i>	Type name
<i>td</i>	Type decortation or id

Definition at line 173 of file ParseTpy.h.

The documentation for this class was generated from the following file:

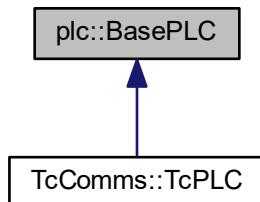
- [ParseTpy.h](#)

8.7 plc::BasePLC Class Reference

Base PLC.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BasePLC:



Public Types

- using `mutex_type` = `std::recursive_mutex`
Defines the mutex type.
- using `guard` = `std::lock_guard< mutex_type >`
Defined the mutex guard type.
- using `time_type` = `DataValueTypeDef::type_uint64`
Define timestamp type.
- using `scanner_func` = `void(BasePLC::*)()`
Function pointer to scanner.

Public Member Functions

- `BasePLC () noexcept`
Default constructor.
- `virtual ~BasePLC ()=default`
Destructor.
- `int get_read_scanner_period () const noexcept`
Get read scanner period in ms.
- `void set_read_scanner_period (int period) noexcept`
Set read scanner period in ms.
- `bool start_read_scanner () noexcept`
Start read scanner.
- `int get_write_scanner_period () const noexcept`
Get write scanner period in ms.
- `void set_write_scanner_period (int period) noexcept`
Set write scanner period in ms.
- `bool start_write_scanner () noexcept`
Start write scanner.
- `int get_update_scanner_period () const noexcept`
Get update scanner period in ms.
- `void set_update_scanner_period (int period) noexcept`
Set update scanner period in ms.
- `bool start_update_scanner () noexcept`
Start update scanner.
- `bool is_scanner_active () const noexcept`
is scanner active?
- `void set_scanners_active (bool active) noexcept`
set scanner active state
- `void reserve (BaseRecordList::size_type n)`
- `bool add (BaseRecord *precord)`
- `bool add (BaseRecordPtr precord)`
- `BaseRecordPtr find (const std::stringcase &name)`
- `bool erase (const std::stringcase &name)`
- `bool get_next (BaseRecordPtr &next, const plc::BaseRecord *prev) const`
- `template<typename func >`
`void for_each (func &f)`
- `template<typename func >`
`void for_each (const func &f)`
- `int count () const noexcept`
Count the number of records.
- `virtual void printAllRecords ()`
Print all records and vals to stdout. (override for action)
- `virtual void printRecord (const std::string &var)`
- `virtual time_type get_timestamp () const noexcept`
Get time stamp.
- `virtual time_t get_timestamp_unix () const noexcept`
- `virtual void set_timestamp (time_type tstamp) noexcept`
Set time stamp.
- `virtual void update_timestamp () noexcept`
Set time stamp to current time.
- `const std::stringcase & get_name () const noexcept`
Get name.

- const `std::stringcase & get_alias () const noexcept`
Get nick name/alias.
- void `set_alias (const std::stringcase &nickname)`
Set nick name/alias.
- virtual bool `start ()`
- virtual void `user_data_set_valid (bool valid) noexcept`
- virtual void `plc_data_set_valid (bool valid) noexcept`

Protected Member Functions

- void `set_name (const std::stringcase &n)`
Set name (careful! This is used for indexing in the PLCList of [System](#))
- virtual void `read_scanner ()`
read scanner (override for action)
- virtual void `write_scanner ()`
write scanner (override for action)
- virtual void `update_scanner ()`
update scanner (override for action)

Protected Attributes

- `mutex_type mux`
Mutex to synchronize access to this class.
- `std::stringcase name`
Name.
- `std::stringcase alias`
Nick name or alias (used to generate info record names)
- `BaseRecordList records`
- `time_type timestamp`
Time stamp.
- int `read_scanner_period`
read scanner period in ms
- int `write_scanner_period`
write scanner period in ms
- int `update_scanner_period`
update scanner period in ms
- `std::atomic< bool > scanners_active`
scanners are active
- `std::thread read_thread`
read thread
- `std::thread write_thread`
write thread
- `std::thread update_thread`
update thread

8.7.1 Detailed Description

Base PLC.

This is a base class for interfacing a programmable logic controller. It contains and manages a list of tag/channel records. This is a base class which needs to be used a derived class by a real implementation.

This class is MT safe and uses a mutex to synchronize access.

Definition at line 721 of file plcBase.h.

8.7.2 Member Function Documentation

8.7.2.1 add() [1/2]

```
bool plc::BasePLC::add (
    BaseRecord * precord ) [inline]
```

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

<i>precord</i>	Pointer to record. Will be adopted
----------------	------------------------------------

Returns

true, if it could be added

Definition at line 779 of file plcBase.h.

8.7.2.2 add() [2/2]

```
bool plc::BasePLC::add (
    BaseRecordPtr precord )
```

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

<i>precord</i>	Smart pointer to record.
----------------	--------------------------

Returns

true, if it could be added

Definition at line 525 of file plcBase.cpp.

References mux, and records.

8.7.2.3 `erase()`

```
bool plc::BasePLC::erase (
    const std::string& name )
```

Erase a tag/channel record.

Parameters

<code>name</code>	Name of record
-------------------	----------------

Returns

true if erased

Definition at line 553 of file plcBase.cpp.

References mux, name, and records.

8.7.2.4 `find()`

```
BaseRecordPtr plc::BasePLC::find (
    const std::string& name )
```

Find a new tag/channel record.

Parameters

<code>name</code>	Name of record
-------------------	----------------

Returns

Smart pointer to record (contains nullptr when not found)

Definition at line 539 of file plcBase.cpp.

References mux, name, and records.

8.7.2.5 `for_each()` [1/2]

```
template<typename func >
void plc::BasePLC::for_each (
    const func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

<code>f</code>	Function which takes <code>BaseRecord*</code> as the argument
----------------	---

Definition at line 230 of file `plcBaseTemplate.h`.

References `mux`, and `records`.

8.7.2.6 `for_each()` [2/2]

```
template<typename func >
void plc::BasePLC::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

<code>f</code>	Function which takes <code>BaseRecord*</code> as the argument
----------------	---

Definition at line 219 of file `plcBaseTemplate.h`.

References `mux`, and `records`.

Referenced by `TcComms::TcPLC::write_scanner()`.

Here is the caller graph for this function:

8.7.2.7 `get_next()`

```
bool plc::BasePLC::get_next (
    BaseRecordPtr & next,
    const plc::BaseRecord * prev ) const
```

Get next record in list. Resets to the beginning, if the list changes between subsequent access. This is an MT safe access method to cycle through the list. However, it is not high performance and is meant for slow external access.

Parameters

<code>next</code>	Next tag/channel record (return)
<code>prev</code>	tag/channel record

Returns

true if successful

Definition at line 562 of file plcBase.cpp.

References plc::BaseRecord::get_name(), mux, and records.

Referenced by TcComms::TcPLC::update_scanner().

Here is the call graph for this function: Here is the caller graph for this function:

8.7.2.8 get_timestamp_unix()

```
time_t plc::BasePLC::get_timestamp_unix ( ) const [virtual], [noexcept]
```

Get time stamp as unix time (seconds since 1970-01-01 00:00:00) Does not include leap seconds

Definition at line 586 of file plcBase.cpp.

References timestamp.

Referenced by InfoPlc::InfoInterface::info_update_timestamp_day(), InfoPlc::InfoInterface::info_update_timestamp_hour(), InfoPlc::InfoInterface::info_update_timestamp_local(), InfoPlc::InfoInterface::info_update_timestamp_min(), InfoPlc::InfoInterface::info_update_timestamp_month(), InfoPlc::InfoInterface::info_update_timestamp_sec(), InfoPlc::InfoInterface::info_update_timestamp_str(), and InfoPlc::InfoInterface::info_update_timestamp_year().

Here is the caller graph for this function:

8.7.2.9 plc_data_set_valid()

```
void plc::BasePLC::plc_data_set_valid (
    bool valid ) [virtual], [noexcept]
```

Set the valid flag for all data values by the plc

Parameters

<i>valid</i>	Valid flag, true for valid, false for invalid
--------------	---

Definition at line 633 of file plcBase.cpp.

References plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:

8.7.2.10 printRecord()

```
virtual void plc::BasePLC::printRecord (
    const std::string & var ) [inline], [virtual]
```

Print a record values to stdout. (override for action)

Parameters

<i>var</i>	variable name (accepts wildcards)
------------	-----------------------------------

Reimplemented in [TcComms::TcPLC](#).

Definition at line 819 of file plcBase.h.

8.7.2.11 reserve()

```
void plc::BasePLC::reserve (
    BaseRecordList::size_type n ) [inline]
```

Reserves the given number of elements in the tag/channel list. Use this function when you know many elements are added beforehand to avoid unnecessary rehashing.

Parameters

<i>n</i>	Number of expected tag/channel records
----------	--

Definition at line 774 of file plcBase.h.

References records.

8.7.2.12 start()

```
virtual bool plc::BasePLC::start () [inline], [virtual]
```

This function is called by the driver support in drvtc.cpp, after the PLC has been initialized. This function is overridden by the derived PLCs, and generally will start all the scanner threads.

Returns

true if successful

Reimplemented in [TcComms::TcPLC](#).

Definition at line 843 of file plcBase.h.

8.7.2.13 user_data_set_valid()

```
void plc::BasePLC::user_data_set_valid (
    bool valid ) [virtual], [noexcept]
```

Set the valid flag for all data values by the user

Parameters

<code>valid</code>	Valid flag, true for valid, false for invalid
--------------------	---

Definition at line 623 of file plcBase.cpp.

References `plc::BaseRecord::UserSetValid()`.

Here is the call graph for this function:

8.7.3 Member Data Documentation

8.7.3.1 records

`BaseRecordList plc::BasePLC::records [protected]`

List of tags/channels. The load factor is initialized to 0.5.

Definition at line 864 of file plcBase.h.

Referenced by `add()`, `BasePLC()`, `count()`, `erase()`, `find()`, `for_each()`, `get_next()`, `TcComms::TcPLC::printAllRecords()`, `TcComms::TcPLC::printRecord()`, `TcComms::TcPLC::read_scanner()`, `reserve()`, and `TcComms::TcPLC::update_scanner()`.

The documentation for this class was generated from the following files:

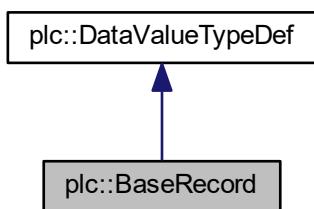
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.8 plc::BaseRecord Class Reference

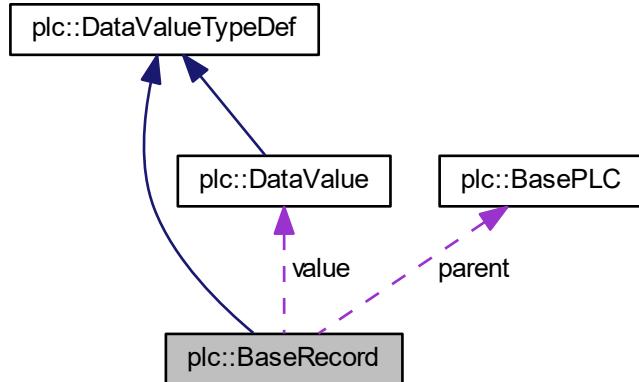
Class for managing a tag/channel.

```
#include <plcBase.h>
```

Inheritance diagram for `plc::BaseRecord`:



Collaboration diagram for plc::BaseRecord:



Public Member Functions

- `BaseRecord () noexcept`
Default constructor.
- `BaseRecord (const std::stringcase &tag) noexcept`
- `BaseRecord (const std::stringcase &recordName, data_type_enum rt, Interface *puser=nullptr, Interface *pplc=nullptr) noexcept`
- `virtual ~BaseRecord ()`
Destructor.
- `const std::stringcase & get_name () const noexcept`
Get name.
- `void set_name (const std::stringcase &recordName) noexcept`
Set name.
- `bool get_process () const noexcept`
Get process flag: false = disabled, true = enabled.
- `void set_process (bool isEnabled) noexcept`
Set process flag.
- `access_rights_enum get_access_rights () noexcept`
Get access rights.
- `void set_access_rights (access_rights_enum rights) noexcept`
Set access rights.
- `time_type get_timestamp () const noexcept`
Get time stamp.
- `virtual Interface * get_userInterface () const noexcept`
Get pointer to user interface (no ownership transfer)
- `virtual Interface * get_plcInterface () const noexcept`
Get pointer to plc interface (no ownership transfer)
- `void set_userInterface (Interface *puser) noexcept`
Set user interface (object will be adopted!)
- `void set_plcInterface (Interface *pplc) noexcept`

- Set plc interface (object will be adopted!)*

 - virtual `BasePLC * get_parent () const noexcept`
Get parent plc.
 - virtual void `set_parent (BasePLC *pPLC) noexcept`
Set parent plc.
 - const `DataValue & get_data () const noexcept`
Get a const reference to the data object.
 - `DataValue & get_data () noexcept`
Get a reference to the data object.
 - bool `DataIsValid () noexcept`
Returns true, if the data is valid.
 - template<typename T >
 bool `UserRead (T &data) noexcept`
 - bool `UserRead (type_string_value *data, size_type max) noexcept`
 - bool `UserRead (DataValue::type_wstring_value *data, size_type max) noexcept`
 - template<typename T >
 bool `UserWrite (const T &data) noexcept`
 - bool `UserWrite (const type_string_value *data, size_type max) noexcept`
 - bool `UserWrite (const type_wstring_value *data, size_type max) noexcept`
 - `size_type UserReadBinary (type_binary p, size_type len) noexcept`
 - `size_type UserWriteBinary (const type_binary p, size_type len) noexcept`
 - bool `UserIsDirty () const noexcept`
Checks if the user needs to read an updated value.
 - void `UserSetDirty () noexcept`
Set dirty flag for user.
 - bool `UserPush (bool force=false) noexcept`
 - bool `UserPull () noexcept`
Pulls the user for new data.
 - void `UserSetValid (bool valid) noexcept`
 - bool `UserGetValid () noexcept`
 - template<typename T >
 bool `PlcRead (T &data) noexcept`
 - bool `PlcRead (type_string_value *data, size_type max) noexcept`
 - bool `PlcRead (type_wstring_value *data, size_type max) noexcept`
 - template<typename T >
 bool `PlcWrite (const T &data) noexcept`
 - bool `PlcWrite (const type_string_value *data, size_type max) noexcept`
 - bool `PlcWrite (const type_wstring_value *data, size_type max) noexcept`
 - `size_type PlcReadBinary (type_binary p, size_type len) noexcept`
 - `size_type PlcWriteBinary (const type_binary p, size_type len) noexcept`
 - bool `PlcIsDirty () const noexcept`
Checks if the plc needs to read an updated value.
 - void `PlcSetDirty () noexcept`
Set dirty flag for plc.
 - bool `PlcPush (bool force=false) noexcept`
 - bool `PlcPull () noexcept`
Pulls the plc for new data.
 - void `PlcSetValid (bool valid) noexcept`
 - bool `PlcGetValid () noexcept`

Protected Attributes

- `std::stringcase name`
Name.
- `access_rights_enum access`
Enum for access rights.
- `atomic_bool process`
Process flag: false = disabled, true = enabled.
- `DataValue value`
Data value.
- `InterfacePtr plc`
PLC interface (master)
- `InterfacePtr user`
User interface (slave)
- `BasePLC * parent`
PLC that this record belongs to.

Additional Inherited Members

8.8.1 Detailed Description

Class for managing a tag/channel.

This is the base class for a tag/channel. It contains a data value and owns two pointers to a user and plc interface, respectively. This is the basic work horse for exchanging data between a user (slave) and a plc (master).

Definition at line 496 of file plcBase.h.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 BaseRecord() [1/2]

```
plc::BaseRecord::BaseRecord (
    const std::stringcase & tag ) [inline], [explicit], [noexcept]
```

Constructor

Parameters

<code>tag</code>	Name of tag/channel
------------------	---------------------

Definition at line 503 of file plcBase.h.

8.8.2.2 BaseRecord() [2/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& recordName,
    data_type_enum rt,
    Interface * puser = nullptr,
    Interface * pplc = nullptr ) [inline], [noexcept]
```

Constructor

Parameters

<i>recordName</i>	Name of tag/channel
<i>rt</i>	Data type
<i>puser</i>	Pointer to user interface object (will be adopted!)
<i>pplc</i>	Pointer to plc interface object (will be adopted!)

Definition at line 510 of file plcBase.h.

8.8.3 Member Function Documentation

8.8.3.1 PlcGetValid()

```
bool plc::BaseRecord::PlcGetValid () [inline], [noexcept]
```

Get the plc valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 683 of file plcBase.h.

References plc::DataValue::PlcGetValid(), PlcPull(), process, and value.

Here is the call graph for this function:

8.8.3.2 PlcPush()

```
bool plc::BaseRecord::PlcPush (
    bool force = false ) [inline], [noexcept]
```

Initiated a plc pull, if the data needs an update

Parameters

<i>force</i>	Forces a plc update even when not needed
--------------	--

Definition at line 193 of file plcBaseTemplate.h.

Referenced by PlcSetDirty(), PlcSetValid(), UserWrite(), and UserWriteBinary().

Here is the caller graph for this function:

8.8.3.3 PlcRead() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcRead (
    T & data ) [inline], [noexcept]
```

Execute a plc read, but pull user first

Parameters

<i>data</i>	Reference to data (return)
-------------	----------------------------

Returns

true if successfull

Definition at line 622 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

Referenced by InfoPlc::InfoInterface::printVal().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.4 PlcRead() [2/3]

```
bool plc::BaseRecord::PlcRead (
    type_string_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a plc read, but pull user first

Parameters

<i>data</i>	character pointer, pchar (return)
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 628 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

Here is the call graph for this function:

8.8.3.5 PlcRead() [3/3]

```
bool plc::BaseRecord::PlcRead (
    type_wstring_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a plc read, but pull user first

Parameters

<i>data</i>	character pointer, pwchar (return)
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 634 of file plcBase.h.

References plc::DataValue::PlcRead(), UserPull(), and value.

Here is the call graph for this function:

8.8.3.6 PlcReadBinary()

```
size_type plc::BaseRecord::PlcReadBinary (
    type_binary p,
    size_type len ) [inline], [noexcept]
```

Execute a plc read, but pull user first

Parameters

<i>p</i>	Pointer to data (destination buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

Definition at line 659 of file plcBase.h.

References plc::DataValue::PlcReadBinary(), UserPull(), and value.

Referenced by TcComms::tcProcWrite::operator()().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.7 PlcSetValid()

```
void plc::BaseRecord::PlcSetValid (
    bool valid ) [inline], [noexcept]
```

Set the plc valid flag and set the dirty flag when flag changes

Parameters

<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 680 of file plcBase.h.

References PlcPush(), plc::DataValue::PlcSetValid(), and value.

Referenced by plc::BasePLC::plc_data_set_valid().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.8 PlcWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcWrite (
    const T & data ) [inline], [noexcept]
```

Execute a plc write and push user

Parameters

<i>data</i>	Reference to data
-------------	-------------------

Returns

true if successfull

Definition at line 640 of file plcBase.h.

References plc::DataValue::PlcWrite(), UserPush(), and value.

Referenced by InfoPlc::InfoInterface::info_update_active(), InfoPlc::InfoInterface::info_update_alias(), InfoPlc::InfoInterface::info_update_name(), InfoPlc::InfoInterface::info_update_rate_read(), InfoPlc::InfoInterface::info_update_rate_update(), InfoPlc::InfoInterface::info_update_rate_write(), InfoPlc::InfoInterface::info_update_records_num(), InfoPlc::InfoInterface::info_update_state(), InfoPlc::InfoInterface::info_update_statestr(), InfoPlc::InfoInterface::info_update_timestamp_day(), InfoPlc::InfoInterface::info_update_timestamp_hour(), InfoPlc::InfoInterface::info_update_timestamp_local(), InfoPlc::InfoInterface::info_update_timestamp_min(), InfoPlc::InfoInterface::info_update_timestamp_month(), InfoPlc::InfoInterface::info_update_timestamp_sec(), InfoPlc::InfoInterface::info_update_timestamp_str(), InfoPlc::InfoInterface::info_update_timestamp_year(), InfoPlc::InfoInterface::info_update_tpy_filename(), and InfoPlc::InfoInterface::info_update_tpy_valid().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.9 PlcWrite() [2/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a plc write and push user

Parameters

<i>data</i>	character pointer, pchar
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 646 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:

8.8.3.10 PlcWrite() [3/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a plc write and push user

Parameters

<i>data</i>	character pointer, pwchar
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 652 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:

8.8.3.11 PlcWriteBinary()

```
size_type plc::BaseRecord::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline], [noexcept]
```

Execute a plc write and push user

Parameters

<i>p</i>	Pointer to data (source buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes written (0 on error)

Definition at line 665 of file plcBase.h.

References `plc::DataValue::PlcWriteBinary()`, `UserPush()`, and `value`.

Referenced by `TcComms::TcPLC::read_scanner()`.

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.12 UserGetValid()

```
bool plc::BaseRecord::UserGetValid () [inline], [noexcept]
```

Get the user valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 616 of file plcBase.h.

References `process`, `plc::DataValue::UserGetValid()`, `UserPull()`, and `value`.

Here is the call graph for this function:

8.8.3.13 UserPush()

```
bool plc::BaseRecord::UserPush (
    bool force = false ) [inline], [noexcept]
```

Initiated a user pull, if the data needs an update

Parameters

<i>force</i>	Forces a user update even when not needed
--------------	---

Definition at line 167 of file plcBaseTemplate.h.

Referenced by `PlcWrite()`, `PlcWriteBinary()`, `UserSetDirty()`, and `UserSetValid()`.

Here is the caller graph for this function:

8.8.3.14 UserRead() [1/3]

```
bool plc::BaseRecord::UserRead (
    DataValue::type_wstring_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	character pointer, pwchar (return)
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 568 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:

8.8.3.15 UserRead() [2/3]

```
template<typename T >
bool plc::BaseRecord::UserRead (
    T & data )  [inline], [noexcept]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	Reference to data (return)
-------------	----------------------------

Returns

true if successfull

Definition at line 556 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Referenced by DevTc::epics_record_traits< RecType >::read().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.16 UserRead() [3/3]

```
bool plc::BaseRecord::UserRead (
    type_string_value * data,
    size_type max )  [inline], [noexcept]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	character pointer, pchar (return)
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 562 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:

8.8.3.17 UserReadBinary()

```
size_type plc::BaseRecord::UserReadBinary (
    type_binary p,
    size_type len ) [inline], [noexcept]
```

Execute a user read, but pull plc first

Parameters

<i>p</i>	Pointer to data (destination buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

Definition at line 592 of file plcBase.h.

References PlcPull(), plc::DataValue::UserReadBinary(), and value.

Here is the call graph for this function:

8.8.3.18 UserSetValid()

```
void plc::BaseRecord::UserSetValid (
    bool valid ) [inline], [noexcept]
```

Set the user valid flag and set the dirty flag when flag changes

Parameters

<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 613 of file plcBase.h.

References UserPush(), plc::DataValue::UserSetValid(), and value.

Referenced by TcComms::TcPLC::read_scanner(), and plc::BasePLC::user_data_set_valid().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.19 UserWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::UserWrite (
    const T & data ) [inline], [noexcept]
```

Execute a user write and push plc

Parameters

<i>data</i>	Reference to data
-------------	-------------------

Returns

true if successfull

Definition at line 573 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Referenced by DevTc::epics_record_traits< RecType >::write().

Here is the call graph for this function: Here is the caller graph for this function:

8.8.3.20 UserWrite() [2/3]

```
bool plc::BaseRecord::UserWrite (
    const type_string_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a user write and push plc

Parameters

<i>data</i>	character pointer, pchar
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 579 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:

8.8.3.21 UserWrite() [3/3]

```
bool plc::BaseRecord::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline], [noexcept]
```

Execute a user write and push plc

Parameters

<i>data</i>	character pointer, pwchar
<i>max</i>	Maximum number of characters

Returns

true if successfull

Definition at line 585 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:

8.8.3.22 UserWriteBinary()

```
size_type plc::BaseRecord::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline], [noexcept]
```

Execute a user write and push plc

Parameters

<i>p</i>	Pointer to data (source buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes written (0 on error)

Definition at line 598 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWriteBinary(), and value.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

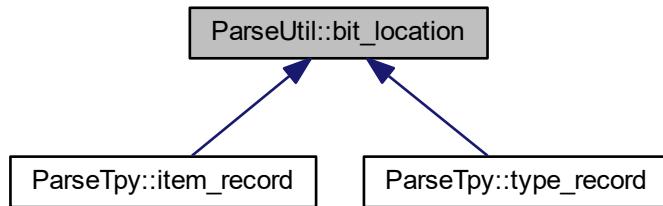
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.9 ParseUtil::bit_location Class Reference

Bit location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::bit_location:



Public Member Functions

- `bit_location () noexcept=default`
Default constructor.
- `bit_location (int bo, int bs) noexcept`
Constructor.
- `bool isValid () const noexcept`
Validity.
- `const int get_bit_offset () const noexcept`
Get bit offset.
- `void set_bit_offset (int ofs) noexcept`
Set bit offset.
- `const int get_bit_size () const noexcept`
Get bit size.
- `void set_bit_size (int size) noexcept`
Set bit size.

Protected Attributes

- `int bitoffs = 0`
bit offset where elements is stored
- `int bitsize = 0`
size in number of bits of symbol

8.9.1 Detailed Description

Bit location.

This is a class for storing bit offset and size in a structure

Definition at line 258 of file ParseUtil.h.

The documentation for this class was generated from the following file:

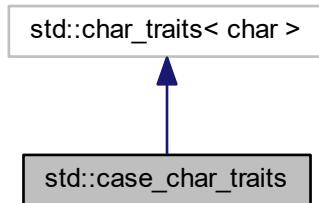
- [ParseUtil.h](#)

8.10 std::case_char_traits Struct Reference

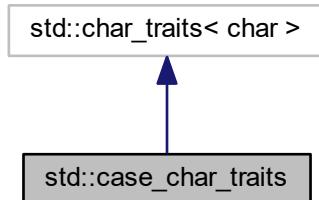
case insensitive traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case_char_traits:



Collaboration diagram for std::case_char_traits:



Static Public Member Functions

- static bool [eq](#) (const char_type &c1, const char_type &c2) noexcept
- static bool [ne](#) (const char_type &c1, const char_type &c2) noexcept
- static bool [lt](#) (const char_type &c1, const char_type &c2) noexcept
- static int [compare](#) (const char_type *s1, const char_type *s2, size_t n) noexcept

8.10.1 Detailed Description

case insensitive traits.

This traits class is not case sensitive.

Definition at line 40 of file stringcase.h.

The documentation for this struct was generated from the following file:

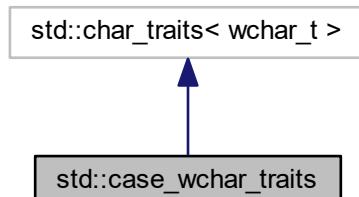
- [stringcase.h](#)

8.11 std::case_wchar_traits Struct Reference

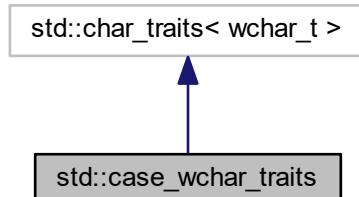
case insensitive unicode traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case_wchar_traits:



Collaboration diagram for std::case_wchar_traits:



Static Public Member Functions

- static bool [eq](#) (const char_type &c1, const char_type &c2) noexcept
- static bool [ne](#) (const char_type &c1, const char_type &c2) noexcept
- static bool [lt](#) (const char_type &c1, const char_type &c2) noexcept
- static int [compare](#) (const char_type *s1, const char_type *s2, size_t n) noexcept

8.11.1 Detailed Description

case insensitive unicode traits.

This unicode traits class is not case sensitive.

Definition at line 72 of file stringcase.h.

The documentation for this struct was generated from the following file:

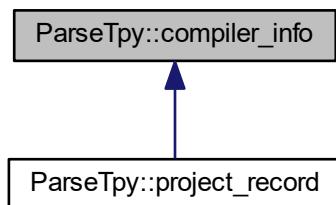
- [stringcase.h](#)

8.12 ParseTpy::compiler_info Class Reference

Compiler information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::compiler_info:



Public Member Functions

- `compiler_info () noexcept=default`
Default constructor.
- `const std::stringcase & get_cmpl_versionstr () const noexcept`
Get compiler version string.
- `void set_cmpl_versionstr (const std::stringcase &versionstr)`
Set compiler version string.
- `double get_cmpl_version () const noexcept`
Get compiler version.
- `const std::stringcase & get_tcat_versionstr () const noexcept`
Get twincat version string.
- `void set_tcat_versionstr (const std::stringcase &versionstr)`
Set twincat version string.
- `unsigned int get_tcat_version_major () const noexcept`
Get twincat major version.
- `unsigned int get_tcat_version_minor () const noexcept`
Get twincat minor version.
- `unsigned int get_tcat_version_build () const noexcept`
Get twincat build version.
- `const std::stringcase & get_cpu_family () const noexcept`
Get cpu family string.
- `void set_cpu_family (const std::stringcase &family)`
Set cpu family string.
- `bool is_cmpl_Valid () const noexcept`
Checks, if version is of the form n.n...
- `bool is_tcat_Valid () const noexcept`
Checks, if twincat version is of the form n.n...

Protected Attributes

- `std::stringcase cmpl_versionstr`
version string
- `double cmpl_version = 0.0`
version number
- `std::stringcase tcat_versionstr`
twincat version string
- `unsigned int tcat_version_major = 0`
twincat major version number
- `unsigned int tcat_version_minor = 0`
twincat minor version number
- `unsigned int tcat_version_build = 0`
twincat build version number
- `std::stringcase cpu_family`
cpu family string

8.12.1 Detailed Description

Compiler information.

This is a base class for storing the compiler information

Definition at line 80 of file ParseTpy.h.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.13 TcComms::DataPar Struct Reference

Memory location struct.

```
#include <tcComms.h>
```

Public Attributes

- unsigned long [indexGroup](#)
index group in ADS server
- unsigned long [indexOffset](#)
index offset in ADS server
- unsigned long [length](#)
count of bytes to read

8.13.1 Detailed Description

Memory location struct.

Struct for storing index group, index offset, and size of a TC symbol

Definition at line 57 of file tcComms.h.

The documentation for this struct was generated from the following file:

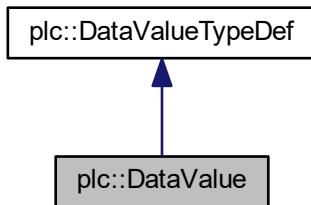
- [tcComms.h](#)

8.14 plc::DataValue Class Reference

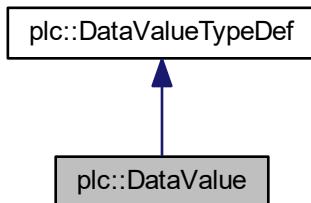
Data value.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValue:



Collaboration diagram for plc::DataValue:



Public Types

- using `data_type` = `void *`
Internally used storage pointer type.

Public Member Functions

- `DataValue () noexcept`
Default constructor.
- `DataValue (data_type_enum rt, size_type len=0) noexcept`
- `~DataValue ()`
Destructor.
- `DataValue (const DataValue &) noexcept`

- Copy constructor.*
- `DataValue & operator= (const DataValue &) noexcept`
- Assignment operator.*
- `void Init (data_type_enum rt, size_type len=0) noexcept`
 - `bool IsValid () const noexcept`
 - is valid*
 - `data_type_enum get_data_type () const noexcept`
 - get type*
 - `size_type get_size () const noexcept`
 - get size*
- template<typename T >
`bool UserRead (T &data) const noexcept`
 - template<size_type N>
`bool UserRead (type_string_value(&data)[N]) const noexcept`
 - `bool UserRead (type_string_value *data, size_type max) const noexcept`
 - `bool UserRead (type_wstring_value *data, size_type max) const noexcept`
 - template<typename T >
`bool UserWrite (const T &data) noexcept`
 - template<size_type N>
`bool UserWrite (const type_string_value(&data)[N]) noexcept`
 - `bool UserWrite (const type_string_value *data, size_type max) noexcept`
 - `bool UserWrite (const type_wstring_value *data, size_type max) noexcept`
 - `size_type UserReadBinary (type_binary p, size_type len) const noexcept`
 - `size_type UserWriteBinary (const type_binary p, size_type len) noexcept`
 - `bool UserIsDirty () const noexcept`
- New data for user.*
- `void UserSetDirty () noexcept`
 - Set dirty flag for user.*
- `void UserSetValid (bool valid) noexcept`
 - `bool UserGetValid () const noexcept`
- template<typename T >
`bool PlcRead (T &data) const noexcept`
 - template<size_type N>
`bool PlcRead (type_string_value(&data)[N]) const noexcept`
 - `bool PlcRead (type_string_value *data, size_type max) const noexcept`
 - `bool PlcRead (type_wstring_value *data, size_type max) const noexcept`
 - template<typename T >
`bool PlcWrite (const T &data) noexcept`
 - template<size_type N>
`bool PlcWrite (const type_string_value(&data)[N]) noexcept`
 - `bool PlcWrite (const type_string_value *data, size_type max) noexcept`
 - `bool PlcWrite (const type_wstring_value *data, size_type max) noexcept`
 - `size_type PlcReadBinary (type_binary p, size_type len) const noexcept`
 - `size_type PlcWriteBinary (const type_binary p, size_type len) noexcept`
 - `bool PlcIsDirty () const noexcept`
- New data for plc.*
- `void PlcSetDirty () noexcept`
 - Set dirty flag for plc.*
- `void PlcSetValid (bool valid) noexcept`
 - `bool PlcGetValid () const noexcept`

Protected Member Functions

- `DataValue (DataValue &&)=delete`
Move constructor (hidden)
- `DataValue & operator= (DataValue &&)=delete`
Move assignment operator.
- template<typename T >
`bool Read (atomic_bool &dirty, T &data) const noexcept`
- `bool Read (atomic_bool &dirty, type_string &data) const noexcept`
- `bool Read (atomic_bool &dirty, type_wstring &data) const noexcept`
- `bool Read (atomic_bool &dirty, type_string_value *data, size_type max) const noexcept`
- `bool Read (atomic_bool &dirty, type_wstring_value *data, size_type max) const noexcept`
- template<typename T >
`bool Write (atomic_bool &dirty, const atomic_bool &pend, const T &data) noexcept`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string &data) noexcept`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring &data) noexcept`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string_value *data, size_type max) noexcept`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring_value *data, size_type max) noexcept`
- `size_type ReadBinary (atomic_bool &dirty, type_binary p, size_type len) const noexcept`
- `size_type WriteBinary (atomic_bool &dirty, const atomic_bool &pend, const type_binary p, size_type len) noexcept`
- `void SetValid (atomic_bool &dirty, bool valid) noexcept`
- `bool GetValid (atomic_bool &dirty) const noexcept`

Protected Attributes

- `data_type mydata`
Data pointer.
- `size_type mysize`
- `data_type_enum mytype`
Data type.
- `atomic_bool myvalid`
Valid flag.
- `atomic_bool myuserdirty`
Dirty flag indicating user needs to update.
- `atomic_bool myplcdirty`
Dirty flag indicating plc needs to update.

Additional Inherited Members

8.14.1 Detailed Description

Data value.

Class for data value This class stores a data value and provides synchronization between the user (slave) and the plc (master) interfaces. When the plc writes a value, it is marked dirty on the user side. Then, when the user reads this data, it resets the dirty flag. The same logic applies for writes by the user and reads by the plc.

Data access is guaranteed to be atomic and MT safe for the simple data types. For strings a mutex is used. If binary data is used, it is not atomic and no mutex is used. Construction, initialization and destruction is not MT safe and all data access has to be stopped during these operations.

Type conversion is provided between all simple data types. However, the loss of information is not checked upon a down cast. Strings have to be read as strings. Binary data needs to be accessed with the binary read/write operations. However, all data can be accessed through binary access. In the later case and for simple data types, the access is atomic.

Definition at line 224 of file plcBase.h.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 DataValue()

```
plc::DataValue::DataValue (
    data_type_enum rt,
    size_type len = 0 ) [inline], [explicit], [noexcept]
```

Constructor

Parameters

<i>rt</i>	Data type enumeration value
<i>len</i>	Length of data

Definition at line 236 of file plcBase.h.

References Init().

Here is the call graph for this function:

8.14.3 Member Function Documentation

8.14.3.1 GetValid()

```
bool plc::DataValue::GetValid (
    atomic_bool & dirty ) const [protected], [noexcept]
```

Get the valid flag and reset the dirty flag

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
--------------	---------------------------------------

Returns

valid True for valid data, False for invalid

Definition at line 490 of file plcBase.cpp.

References `plc::DataValueTypeDef::memory_order`.

Referenced by `PlcGetValid()`, and `UserGetValid()`.

Here is the caller graph for this function:

8.14.3.2 Init()

```
void plc::DataValue::Init (
    data_type_enum rt,
    size_type len = 0 ) [noexcept]
```

Initializes data value

Parameters

<code>rt</code>	Data type enumeration value
<code>len</code>	Length of data (use only for binary)

Definition at line 157 of file plcBase.cpp.

References `plc::dtBinary`, `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `plc::dtWString`, and `plc::DataValue::DataValueTypeDef::memory_order`.

Referenced by `DataValue()`, and `~DataValue()`.

Here is the caller graph for this function:

8.14.3.3 PlcGetValid()

```
bool plc::DataValue::PlcGetValid () const [inline], [noexcept]
```

Get the valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 376 of file plcBase.h.

References `GetValid()`, and `myuserdirty`.

Referenced by `plc::BaseRecord::PlcGetValid()`.

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.4 PlcRead() [1/4]

```
template<typename T >
bool plc::DataValue::PlcRead (
    T & data ) const [inline], [noexcept]
```

Read data by the plc

Parameters

<i>data</i>	Data value reference (return)
-------------	-------------------------------

Definition at line 321 of file plcBase.h.

References myplcdirty, and Read().

Referenced by plc::BaseRecord::PlcRead().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.5 PlcRead() [2/4]

```
bool plc::DataValue::PlcRead (
    type_string_value * data,
    size_type max ) const [inline], [noexcept]
```

Read character array (pchar) by the plc

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 330 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:

8.14.3.6 PlcRead() [3/4]

```
template<size_type N>
bool plc::DataValue::PlcRead (
    type_string_value(&) data[N] ) const [inline], [noexcept]
```

Read fixed length character array data by the plc

Parameters

<i>data</i>	Data value reference for a fixed length character array (return)
-------------	--

Definition at line 325 of file plcBase.h.

References myplcdirty, and ReadBinary().

Here is the call graph for this function:

8.14.3.7 PlcRead() [4/4]

```
bool plc::DataValue::PlcRead (
    type_wstring_value * data,
    size_type max ) const [inline], [noexcept]
```

Read character array (pwchar) by the plc

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 335 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:

8.14.3.8 PlcReadBinary()

```
size_type plc::DataValue::PlcReadBinary (
    type_binary p,
    size_type len ) const [inline], [noexcept]
```

Read data as binary by the plc

Parameters

<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 359 of file plcBase.h.

References myplcdirty, and ReadBinary().

Referenced by plc::BaseRecord::PlcReadBinary().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.9 PlcSetValid()

```
void plc::DataValue::PlcSetValid (
    bool valid ) [inline], [noexcept]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 373 of file plcBase.h.

References myplcdirty, and SetValid().

Referenced by plc::BaseRecord::PlcSetValid().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.10 PlcWrite() [1/4]

```
template<typename T >
bool plc::DataValue::PlcWrite (
    const T & data )  [inline], [noexcept]
```

Write data by the plc

Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 339 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::PlcWrite().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.11 PlcWrite() [2/4]

```
bool plc::DataValue::PlcWrite (
    const type_string_value * data,
    size_type max )  [inline], [noexcept]
```

Write character array (pchar) by the plc

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 348 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

8.14.3.12 PlcWrite() [3/4]

```
template<size_type N>
bool plc::DataValue::PlcWrite (
    const type_string_value(&) data[N] )  [inline], [noexcept]
```

Write fixed length character array data by the plc

Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 343 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:

8.14.3.13 PlcWrite() [4/4]

```
bool plc::DataValue::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline], [noexcept]
```

Write character array (wchar) by the plc

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 353 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

8.14.3.14 PlcWriteBinary()

```
size_type plc::DataValue::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline], [noexcept]
```

Write data as binary by the plc

Parameters

<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 364 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::PlcWriteBinary().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.15 Read() [1/5]

```
template<typename T >
bool plc::DataValue::Read (
    atomic_bool & dirty,
    T & data ) const [protected], [noexcept]
```

Read data

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

[DataValue::Read](#) (bool, Inegral and floating point types)

Definition at line 101 of file plcBaseTemplate.h.

References plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, and plc::reset_and_read().

Referenced by PlcRead(), and UserRead().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.16 Read() [2/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string & data ) const [protected], [noexcept]
```

Read string (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

Definition at line 255 of file plcBase.cpp.

References plc::dtString, and plc::reset_and_read().

Here is the call graph for this function:

8.14.3.17 Read() [3/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string_value * data,
    size_type max ) const [protected], [noexcept]
```

Read character array (pchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 292 of file plcBase.cpp.

8.14.3.18 Read() [4/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring & data ) const [protected], [noexcept]
```

Read wstring (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

Definition at line 272 of file plcBase.cpp.

References plc::dtString, plc::dtWString, and plc::reset_and_read().

Here is the call graph for this function:

8.14.3.19 Read() [5/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring_value * data,
    size_type max ) const [protected], [noexcept]
```

Read character array (pwchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 306 of file plcBase.cpp.

8.14.3.20 ReadBinary()

```
DataValue::size_type plc::DataValue::ReadBinary (
    atomic_bool & dirty,
    type_binary p,
    size_type len ) const [protected], [noexcept]
```

Read data as binary

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 376 of file plcBase.cpp.

References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, and plc::DataValueTypeDef::memory_order.

Referenced by PlcRead(), PlcReadBinary(), UserRead(), and UserReadBinary().

Here is the caller graph for this function:

8.14.3.21 SetValid()

```
void plc::DataValue::SetValid (
    atomic_bool & dirty,
    bool valid ) [protected], [noexcept]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>valid</i>	True for valid data, False for invalid

Definition at line 478 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order.

Referenced by PlcSetValid(), and UserSetValid().

Here is the caller graph for this function:

8.14.3.22 UserGetValid()

```
bool plc::DataValue::UserGetValid () const [inline], [noexcept]
```

Get the valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 317 of file plcBase.h.

References GetValid(), and myplcdirty.

Referenced by plc::BaseRecord::UserGetValid().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.23 UserRead() [1/4]

```
template<typename T >
bool plc::DataValue::UserRead (
    T & data ) const [inline], [noexcept]
```

Read data by the user

Parameters

<i>data</i>	Data value reference (return)
-------------	-------------------------------

Definition at line 261 of file plcBase.h.

References myuserdirty, and Read().

Referenced by plc::BaseRecord::UserRead().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.24 UserRead() [2/4]

```
bool plc::DataValue::UserRead (
    type_string_value * data,
    size_type max ) const [inline], [noexcept]
```

Read character array (pchar) by the user

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 270 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:

8.14.3.25 UserRead() [3/4]

```
template<size_type N>
bool plc::DataValue::UserRead (
    type_string_value(&) data[N] ) const [inline], [noexcept]
```

Read fixed length character array data by the user

Parameters

<i>data</i>	Data value reference for a fixed length character array (return)
-------------	--

Definition at line 265 of file plcBase.h.

References myuserdirty, and ReadBinary().

Here is the call graph for this function:

8.14.3.26 UserRead() [4/4]

```
bool plc::DataValue::UserRead (
    type_wstring_value * data,
    size_type max ) const [inline], [noexcept]
```

Read character array (pwchar) by the user

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 275 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:

8.14.3.27 UserReadBinary()

```
size_type plc::DataValue::UserReadBinary (
    type_binary p,
    size_type len ) const [inline], [noexcept]
```

Read data as binary by the user

Parameters

<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 300 of file plcBase.h.

References myuserdirty, and ReadBinary().

Referenced by plc::BaseRecord::UserReadBinary().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.28 UserSetValid()

```
void plc::DataValue::UserSetValid (
    bool valid ) [inline], [noexcept]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 314 of file plcBase.h.

References myuserdirty, and SetValid().

Referenced by plc::BaseRecord::UserSetValid().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.29 UserWrite() [1/4]

```
template<typename T >
bool plc::DataValue::UserWrite (
    const T & data ) [inline], [noexcept]
```

Write data by the user

Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 279 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::UserWrite().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.30 UserWrite() [2/4]

```
bool plc::DataValue::UserWrite (
    const type_string_value * data,
    size_type max ) [inline], [noexcept]
```

Write character array (pchar) by the user

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 289 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

8.14.3.31 UserWrite() [3/4]

```
template<size_type N>
bool plc::DataValue::UserWrite (
    const type_string_value(&) data[N] ) [inline], [noexcept]
```

Write fixed length character array data by the user

Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 283 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:

8.14.3.32 UserWrite() [4/4]

```
bool plc::DataValue::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline], [noexcept]
```

Write character array (wpchar) by the user

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 294 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

8.14.3.33 UserWriteBinary()

```
size_type plc::DataValue::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline], [noexcept]
```

Write data as binary by the user

Parameters

<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 305 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::UserWriteBinary().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.34 Write() [1/5]

```
template<typename T >
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const T & data ) [protected], [noexcept]
```

Write data

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference (return)

[DataValue::UserWrite](#) (bool, Integral and floating point types)

Definition at line 134 of file plcBaseTemplate.h.

References plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, and plc::write_and_test().

Referenced by PlcWrite(), and UserWrite().

Here is the call graph for this function: Here is the caller graph for this function:

8.14.3.35 Write() [2/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_string & data ) [protected], [noexcept]
```

Write string (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference

Definition at line 320 of file plcBase.cpp.

References `plc::dtString`, `plc::dtWString`, and `plc::write_and_test()`.

Here is the call graph for this function:

8.14.3.36 Write() [3/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_string_value * data,
    size_type max ) [protected], [noexcept]
```

Write character array (pchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 349 of file plcBase.cpp.

8.14.3.37 Write() [4/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring & data ) [protected], [noexcept]
```

Write wstring (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference

Definition at line 336 of file plcBase.cpp.

References `plc::dtWString`, and `plc::write_and_test()`.

Here is the call graph for this function:

8.14.3.38 Write() [5/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring_value * data,
    size_type max ) [protected], [noexcept]
```

Write character array (pwchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 362 of file `plcBase.cpp`.

8.14.3.39 WriteBinary()

```
DataValue::size_type plc::DataValue::WriteBinary (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_binary p,
    size_type len ) [protected], [noexcept]
```

Write data as binary

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 426 of file `plcBase.cpp`.

References `plc::dtBinary`, `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `plc::dtWString`, and `plc::DataValueTypeDef::memory_order`.

Referenced by `PlcWrite()`, `PlcWriteBinary()`, `UserWrite()`, and `UserWriteBinary()`.

Here is the caller graph for this function:

8.14.4 Member Data Documentation

8.14.4.1 `mysize`

`size_type` `plc::DataValue::mysize` [protected]

Size of allocated memory for simple types; size of string class for strings and size of data for binary

Definition at line 466 of file plcBase.h.

Referenced by `get_size()`, and `IsValid()`.

The documentation for this class was generated from the following files:

- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.15 `plc::DataValueTraits< T >` Struct Template Reference

Data value traits.

```
#include <plcBase.h>
```

Public Types

- using `size_type` = `size_t`
size type
- using `data_type_enum` = `plc::data_type_enum`
enumerated type for data type
- using `traits_type` = `T`
traits type
- using `traits_atomic` = `typename std::atomic< T >`
atomic variable type

8.15.1 Detailed Description

```
template<typename T>
struct plc::DataValueTraits< T >
```

Data value traits.

Traits class for data value

Definition at line 112 of file plcBase.h.

The documentation for this struct was generated from the following file:

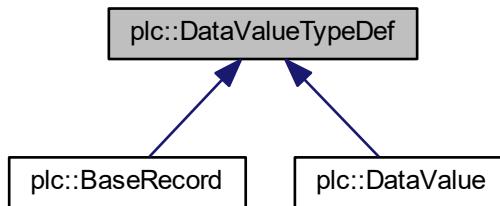
- [plcBase.h](#)

8.16 plc::DataValueTypeDef Struct Reference

Collection of type definitions.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValueTypeDef:



Public Types

- using `size_type` = `size_t`
size type
- using `data_type_enum` = `plc::data_type_enum`
enumerated type for data type
- using `type_bool` = `bool`
bool type
- using `type_int8` = `signed char`
1-byte integer type
- using `type_uint8` = `unsigned char`
1-byte unsigned integer type
- using `type_int16` = `short`
3-byte integer type
- using `type_uint16` = `unsigned short`
3-byte unsigned integer type
- using `type_int32` = `int`
4-byte integer type
- using `type_uint32` = `unsigned int`
4-byte unsigned integer type
- using `type_int64` = `long long`
8-byte integer type
- using `type_uint64` = `unsigned long long`
8-byte unsigned integer type
- using `type_float` = `float`
4-byte single precision floating point type
- using `type_double` = `double`
4-byte double precision floating point type
- using `type_string` = `std::string`

- `string type`
- using `type_wstring` = `std::wstring`
`wstring type`
- using `type_binary` = `void *`
`binary type`
- using `type_string_value` = `std::string::value_type`
`string character type`
- using `type_wstring_value` = `std::wstring::value_type`
`wstring character type`
- using `atomic_bool` = `DataValueTraits< type_bool >::traits_atomic`
`atomic bool type`
- using `atomic_int8` = `DataValueTraits< type_int8 >::traits_atomic`
`atomic 1-byte integer type`
- using `atomic_uint8` = `DataValueTraits< type_uint8 >::traits_atomic`
`atomic 1-byte unsigned integer type`
- using `atomic_int16` = `DataValueTraits< type_int16 >::traits_atomic`
`atomic 2-byte integer type`
- using `atomic_uint16` = `DataValueTraits< type_uint16 >::traits_atomic`
`atomic 2-byte unsigned integer type`
- using `atomic_int32` = `DataValueTraits< type_int32 >::traits_atomic`
`atomic 4-byte integer type`
- using `atomic_uint32` = `DataValueTraits< type_uint32 >::traits_atomic`
`atomic 4-byte unsigned integer type`
- using `atomic_int64` = `DataValueTraits< type_int64 >::traits_atomic`
`atomic 8-byte integer type`
- using `atomic_uint64` = `DataValueTraits< type_uint64 >::traits_atomic`
`atomic 8-byte unsigned integer type`
- using `atomic_float` = `DataValueTraits< type_float >::traits_atomic`
`atomic 4-byte single precision floating point type`
- using `atomic_double` = `DataValueTraits< type_double >::traits_atomic`
`atomic 8-byte double precision floating point type`
- using `atomic_string` = `DataValueTraits< type_string >::traits_atomic`
`atomic string type`
- using `atomic_wstring` = `DataValueTraits< type_wstring >::traits_atomic`
`atomic wstring type`
- using `atomic_binary` = `DataValueTraits< type_binary >::traits_atomic`
`atomic binary type`
- using `time_type` = `DataValueTypeDef::type_uint64`
`Define timestamp type.`

Static Public Attributes

- static const `std::memory_order memory_order` = `std::memory_order_seq_cst`
`memory order used for atomic access`

8.16.1 Detailed Description

Collection of type definitions.

Type definitions for data value

Definition at line 127 of file plcBase.h.

The documentation for this struct was generated from the following file:

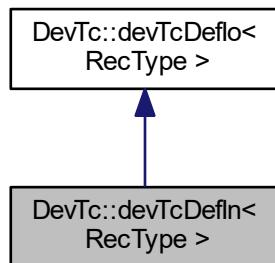
- [plcBase.h](#)

8.17 DevTc::devTcDefIn< RecType > Struct Template Reference

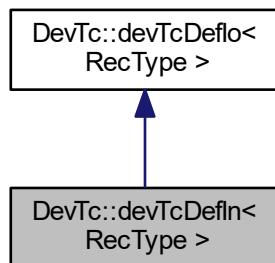
Device support input record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefIn< RecType >:



Collaboration diagram for DevTc::devTcDefIn< RecType >:



Public Types

- using `rec_type = devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
- using `rec_type_ptr = devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.

Public Member Functions

- `devTcDefIn () noexcept`
Constructor.

Static Public Member Functions

- static long `init_read_record (rec_type_ptr prec) noexcept`
init callback for read records
- static long `read (rec_type_ptr precord) noexcept`
read callback

Additional Inherited Members

8.17.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefIn< RecType >
```

Device support input record.

Device Support Record for TwinCAT/ADS input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 390 of file devTc.h.

The documentation for this struct was generated from the following file:

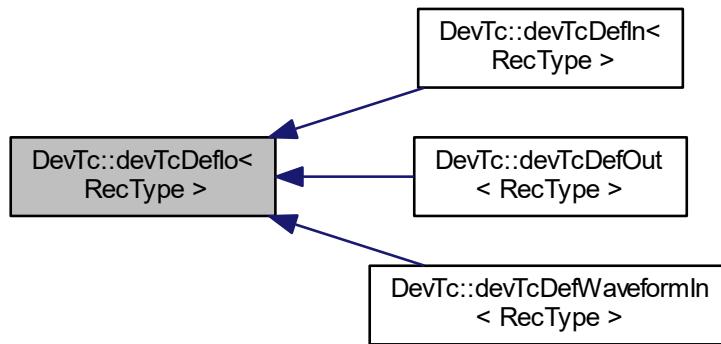
- `devTc.h`

8.18 DevTc::devTcDeflo< RecType > Struct Template Reference

Device support record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDeflo< RecType >:



Public Types

- using `rec_type = epics_record_traits< RecType >::traits_type`
Record type: aiRecord, etc.
- using `rec_type_ptr = epics_record_traits< RecType >::traits_type *`
Pointer to record type.
- using `report_type = long(*)(int level)`
Report support function.
- using `init_type = long(*)(int after)`
Initialization function.
- using `init_record_type = long(*)(rec_type_ptr prec)`
Initialization function.
- using `get_ioint_info_type = long(*)(int cmd, rec_type_ptr prec, IOSCANPVT *piosl)`
IO/INT support function.
- using `io_type = long(*)(rec_type_ptr prec)`
IO function like read or write.
- using `special_linconv_type = long(*)(rec_type_ptr prec, int after)`
Linear conversion support function.

Public Attributes

- long `number` = 6
Number of support functions.
- `report_type report_fn` = nullptr
Report support function.
- `init_type init_fn` = nullptr
Init support function.
- `init_record_type init_record_fn` = nullptr
Record init support function.
- `get_ioint_info_type get_ioint_info_fn` = `get_ioint_info`
IO/INT support function.
- `io_type io_fn` = nullptr
Read/write support function.
- `special_linconv_type special_linconv_fn` = nullptr
Linear conversion support function.

Protected Member Functions

- `devTcDeflo (init_record_type ioinit, io_type io) noexcept`
Constructor for IO record.

Static Protected Member Functions

- static long `get_ioint_info (int cmd, rec_type_ptr prec, IOSCANPVT *ppv)` noexcept
IO/INT info callback.

8.18.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDeflo< RecType >
```

Device support record.

Deviced Support Record for generic TwinCAT/ADS IO This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 340 of file devTc.h.

The documentation for this struct was generated from the following file:

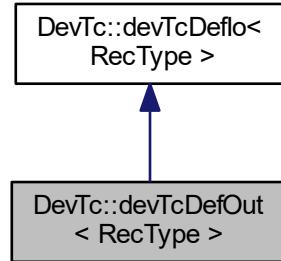
- `devTc.h`

8.19 DevTc::devTcDefOut< RecType > Struct Template Reference

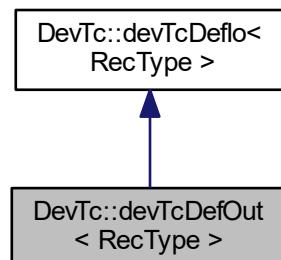
device support output record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefOut< RecType >:



Collaboration diagram for DevTc::devTcDefOut< RecType >:



Public Types

- using `rec_type = devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
- using `rec_type_ptr = devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.

Public Member Functions

- `devTcDefOut () noexcept`
Constructor.

Static Public Member Functions

- static long `init_write_record` (`rec_type_ptr` prec) noexcept
init callback for write records
- static long `write` (`rec_type_ptr` precord) noexcept
write callback

Additional Inherited Members

8.19.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefOut< RecType >
```

device support output record.

Device Support Record for TwinCAT/ADS output This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 412 of file devTc.h.

The documentation for this struct was generated from the following file:

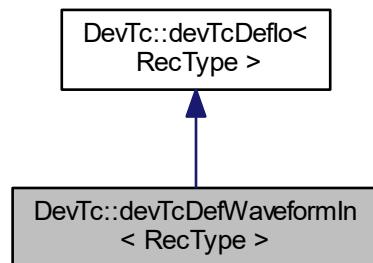
- `devTc.h`

8.20 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference

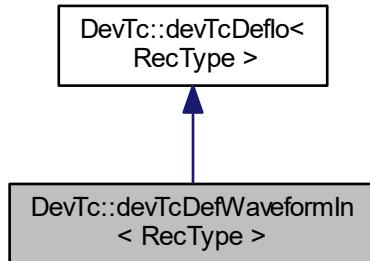
device support waveform record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefWaveformIn< RecType >:



Collaboration diagram for DevTc::devTcDefWaveformIn< RecType >:



Public Types

- using `rec_type = devTcDefflo< RecType >::rec_type`
Record type: aiRecord, etc.
- using `rec_type_ptr = devTcDefflo< RecType >::rec_type_ptr`
Pointer to record type.

Public Member Functions

- `devTcDefWaveformIn () noexcept`
Constructor.

Static Public Member Functions

- static long `init_read_waveform_record (rec_type_ptr prec) noexcept`
init callback for read records
- static long `read_waveform (rec_type_ptr precord) noexcept`
read callback

Additional Inherited Members

8.20.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefWaveformIn< RecType >
```

device support waveform record.

Device Support Record for TwinCAT/ADS waveform input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 434 of file devTc.h.

The documentation for this struct was generated from the following file:

- `devTc.h`

8.21 EpicsTpy::epics_conversion Class Reference

Epics conversion.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_conversion:

Collaboration diagram for EpicsTpy::epics_conversion:

Public Member Functions

- `epics_conversion () noexcept=default`
Default constructor.
- `epics_conversion (case_type caseconv, bool noindex) noexcept`
- `epics_conversion (tc_epics_conv epics_conv, case_type caseconv, bool noldot, bool noindex) noexcept`
- `epics_conversion (int argc, const char *const argv[], bool argp[] = 0)`
- `virtual ~epics_conversion ()=default`
Destructor.
- `virtual int getopt (int argc, const char *const argv[], bool argp[] = 0)`
- `tc_epics_conv get_conversion_rule () const noexcept`
Get the conversion rule.
- `void set_conversion_rule (tc_epics_conv epics_conv) noexcept`
Set the conversion rule.
- `case_type get_case_rule () const noexcept`
Get the conversion rule.
- `void set_case_rule (case_type epics_conv) noexcept`
Set the conversion rule.
- `bool get_dot_rule () const noexcept`
Get the leadin dot rule.
- `void set_dot_rule (bool noldot) noexcept`
Set the leading dot rule.
- `bool get_array_rule () const noexcept`
Get the array index rule.
- `void set_array_rule (bool noindex) noexcept`
Set the array conversion rule.
- `std::stringcase get_prefix () const`
Get the channel prefix.
- `void set_prefix (const std::stringcase &pvPrefix)`
Set the channel prefix.
- `virtual std::string to_epics (const std::stringcase &name, bool published=true, const ParseUtil::substitution **subst=nullptr) const`

Protected Attributes

- `tc_epics_conv conv_rule = tc_epics_conv::ligo_std`
Conversion rule.
- `case_type case_epics_names = case_type::upper`
Case conversion rule.
- `bool no_leading_dot = true`
Leading dot conversion rule.
- `bool no_array_index = true`
Array index conversion rule.
- `std::stringcase prefix`
Prefix to apply to all EPICS names.

Additional Inherited Members

8.21.1 Detailed Description

Epics conversion.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 59 of file TpyToEpics.h.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 epics_conversion() [1/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    case_type caseconv,
    bool noindex ) [inline], [noexcept]
```

Constructor

Parameters

<i>caseconv</i>	Case conversion specification
<i>noindex</i>	Eliminate array indices '[n]' with '_n'

Definition at line 67 of file TpyToEpics.h.

8.21.2.2 epics_conversion() [2/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    tc_epics_conv epics_conv,
    case_type caseconv,
    bool noldot,
    bool noindex ) [inline], [noexcept]
```

Constructor

Parameters

<i>epics_conv</i>	Epics conversion rule
<i>caseconv</i>	Case conversion specification
<i>noldot</i>	Eliminate leading dot in a name
<i>noindex</i>	Eliminate array indices '[n]' with '_n'

Definition at line 74 of file TpyToEpics.h.

8.21.2.3 epics_conversion() [3/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 85 of file TpyToEpics.h.

References getopt().

Here is the call graph for this function:



8.21.3 Member Function Documentation

8.21.3.1 getopt()

```
int EpicsTpy::epics_conversion::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [virtual]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be

ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/fs 'filename': Reads a substitution file and keeps defined or published channels /fi 'filename': Reads a substitution file and keeps only defined channels /fa 'filename': Reads a substitution file and keeps all channels /nd: Eliminates leading dot in channel name (default) /yd: Leaves leading dot in channel name /rn: Does not apply any special conversion rules /rd: Replaces dots with underscores in channel names /rl: LIGO standard conversion rule (default) /rv: LIGO vacuum channel conversion rule /cp: Preserve case in EPICS channel names /cu: Force upper case in EPICS channel names (default) /cl: Force lower case in EPICS channel names /ni: Replaces array brackets with underscore (default) /yi: Leave array indices as is /p 'name': Include a prefix of 'name' for every channel (defaults to no prefix)

The channel name conversions are processed in the order specified above. Replacement rules, if specified, are applied after the substitution file has been processed and before all other steps.

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Reimplemented in [EpicsTpy::epics_macros_processing](#), [EpicsTpy::epics_list_processing](#), [DevTc::epics_tc_db_processing](#), and [EpicsTpy::epics_db_processing](#).

Definition at line 26 of file TpyToEpics.cpp.

Referenced by `epics_conversion()`, `EpicsTpy::epics_list_processing::getopt()`, `EpicsTpy::epics_macros_processing::getopt()`, and `EpicsTpy::epics_db_processing::getopt()`.

Here is the caller graph for this function:

8.21.3.2 to_epics()

```
string EpicsTpy::epics_conversion::to_epics (
    const std::string& name,
    bool published = true,
    const ParseUtil::substitution ** subst = nullptr) const [virtual]
```

Converts a TwinCAT or OPC name to an EPICS channel name

Parameters

<i>name</i>	TwinCAT/opc name
<i>published</i>	This variable is published
<i>subst</i>	Optional substitution return, when non null

Returns

EPICS name

Definition at line 139 of file TpyToEpics.cpp.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_macrofiles_processing::operator()().

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.22 EpicsTpy::epics_db_processing Class Reference

pics database record processing

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_db_processing:

Collaboration diagram for EpicsTpy::epics_db_processing:

Public Member Functions

- [`epics_db_processing \(\) noexcept=default`](#)
Default constructor.
- [`epics_db_processing \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`virtual int getopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`int my getopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`device_support_type get_device_support \(\) const noexcept`](#)
Get device support conversion rule.
- [`void set_device_support \(device_support_type devsup\) noexcept`](#)
Set device support conversion rule.
- [`string_support_type get_string_support \(\) const noexcept`](#)
Get string support conversion rule.
- [`void set_string_support \(string_support_type strsup\) noexcept`](#)
Set string support conversion rule.
- [`int_support_type get_int_support \(\) const noexcept`](#)
Get integer support conversion rule.
- [`void set_int_support \(int_support_type intsup\) noexcept`](#)
Set integer support conversion rule.
- [`virtual bool operator\(\) \(const ParseUtil::process_arg &arg\) noexcept`](#)

Protected Member Functions

- bool `process_field_string (std::stringcase name, std::stringcase val, int maxlen=256) noexcept`
- bool `process_field_numeric (std::stringcase name, int val) noexcept`
- bool `process_field_numeric (std::stringcase name, double val) noexcept`
- bool `process_field_numeric (std::stringcase name, std::stringcase val) noexcept`
- bool `process_field_alarm (std::stringcase name, std::stringcase severity) noexcept`

Protected Attributes

- `device_support_type device_support = device_support_type::tc_name`
Device support field conversion rule.
- `string_support_type string_support = string_support_type::vary_string`
String support field conversion rule.
- `int_support_type int_support = int_support_type::int_auto`
Integer support field conversion rule.

Additional Inherited Members

8.22.1 Detailed Description

pics database record processing

Class for generatig an EPICS database record

Definition at line 722 of file TpyToEpics.h.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 epics_db_processing()

```
EpicsTpy::epics_db_processing::epics_db_processing (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with `epics_conversion:: getopt`, `split_io_support:: getopt` and `mygetopt()`.

Parameters

<code>fname</code>	Output filename
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in <code>main()</code>
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 1256 of file TpyToEpics.cpp.

References getopt(), and EpicsTpy::tc_name.

Here is the call graph for this function:

8.22.3 Member Function Documentation

8.22.3.1 getopt()

```
int EpicsTpy::epics_db_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [virtual]
```

Parse a command line Processed options with [epics_conversion::getopt](#) and mygetopt.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Reimplemented from [EpicsTpy::epics_conversion](#).

Reimplemented in [DevTc::epics_tc_db_processing](#).

Definition at line 1271 of file TpyToEpics.cpp.

References EpicsTpy::epics_conversion::getopt(), and mygetopt().

Referenced by epics_db_processing().

Here is the call graph for this function: Here is the caller graph for this function:

8.22.3.2 mygetopt()

```
int EpicsTpy::epics_db_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be

ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/devopc: Uses OPC name in INPUT/OUTPUT field (default) /devtc: Uses TwinCAT name in INPUT/OUTPUT fields instead of OPC

/ss: Use standard strings (stringin/stringout) /sl: Use long strings (lsi/lso) /sd: Use long or short strings depending on size (default)

/is: Use standard 32-bit integers (longin/longout) /il: Use 64-bit integers (int64in/int64out) /id: Use 32 or 64-bit integers depending on size (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Definition at line 1281 of file TpyToEpics.cpp.

References EpicsTpy::long_string, EpicsTpy::opc_name, set_device_support(), set_string_support(), EpicsTpy::short_string, and EpicsTpy::tc_name.

Referenced by getopt().

Here is the call graph for this function: Here is the caller graph for this function:

8.22.3.3 operator()()

```
bool EpicsTpy::epics_db_processing::operator() (
    const ParseUtil::process_arg & arg ) [virtual], [noexcept]
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successfully processed

Reimplemented in [DevTc::epics_tc_db_processing](#).

Definition at line 1338 of file TpyToEpics.cpp.

References ParseUtil::substitution::get_opc(), ParseUtil::substitution::has_valid_opc(), ParseUtil::opc_list::is_published(), ParseUtil::opc_list::is_READONLY(), and EpicsTpy::MAX_EPICS_CHANNEL.

Here is the call graph for this function:

8.22.3.4 process_field_alarm()

```
bool EpicsTpy::epics_db_processing::process_field_alarm (
    std::stringcase name,
    std::stringcase severity ) [protected], [noexcept]
```

Process a record field of type alarm

Parameters

<i>name</i>	Name of field
<i>severity</i>	Severity of alarm

Returns

True if successful

Definition at line 1735 of file TpyToEpics.cpp.

References EpicsTpy::EPICS_DB_MAJOR, EpicsTpy::EPICS_DB_MINOR, and EpicsTpy::EPICS_DB_NOALARMS.

8.22.3.5 process_field_numeric() [1/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    double val ) [protected], [noexcept]
```

Process a record field of numeric type

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

True if successful

Definition at line 1709 of file TpyToEpics.cpp.

8.22.3.6 process_field_numeric() [2/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    int val )  [protected], [noexcept]
```

Process a record field of numeric type

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

True if successful

Definition at line 1699 of file TpyToEpics.cpp.

8.22.3.7 process_field_numeric() [3/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    std::stringcase val )  [protected], [noexcept]
```

Process a record field of type string

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

True if successful

Definition at line 1719 of file TpyToEpics.cpp.

8.22.3.8 process_field_string()

```
bool EpicsTpy::epics_db_processing::process_field_string (
    std::stringcase name,
    std::stringcase val,
    int maxlen = 256 )  [protected], [noexcept]
```

Process a record field of type string

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field
<i>maxlen</i>	Maximum length of value field

Returns

True if successful

Definition at line 1689 of file TpyToEpics.cpp.

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.23 EpicsTpy::epics_list_processing Class Reference

List processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_list_processing:

Collaboration diagram for EpicsTpy::epics_list_processing:

Public Member Functions

- [`epics_list_processing \(\) noexcept=default`](#)
Default constructor.
- [`epics_list_processing \(listing_type ltype, bool ll=false\) noexcept`](#)
- [`epics_list_processing \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`virtual ~epics_list_processing \(\)=default`](#)
Default destructor.
- [`int getopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\) override`](#)
- [`int mygetopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)`](#)
- [`virtual bool operator\(\) \(const ParseUtil::process_arg &arg\)`](#)
- [`listing_type get_listing \(\) const noexcept`](#)
Get listing type.
- [`void set_listing \(listing_type lt\) noexcept`](#)
Set listing.
- [`bool is_verbose \(\) const noexcept`](#)
Is long listing?
- [`void set_verbose \(bool vrbs\) noexcept`](#)
Set long listing.

Protected Attributes

- `listing_type listing = listing_type::standard`
Listing type.
- `bool verbose = false`
long listing

Additional Inherited Members

8.23.1 Detailed Description

List processing.

Class for generatig a channel list

Definition at line 431 of file TpyToEpics.h.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 epics_list_processing() [1/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    listing_type ltype,
    bool ll = false ) [inline], [explicit], [noexcept]
```

Constructor

Parameters

<code>ltype</code>	Type of listing
<code>//</code>	long listing

Definition at line 439 of file TpyToEpics.h.

8.23.2.2 epics_list_processing() [2/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    const std::string& fname,
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with `epics_conversion::getopt`, `split_io_support::getopt` and `my getopt()`.

Parameters

<i>fname</i>	Output filename
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 573 of file TpyToEpics.cpp.

References [my getopt\(\)](#), and [ParseUtil::standard](#).

Here is the call graph for this function:

8.23.3 Member Function Documentation

8.23.3.1 getopt()

```
int EpicsTpy::epics_list_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [override], [virtual]
```

Parse a command line Processed options with [epics_conversion::getopt](#) and [my getopt\(\)](#).

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Reimplemented from [EpicsTpy::epics_conversion](#).

Definition at line 586 of file TpyToEpics.cpp.

References [EpicsTpy::epics_conversion::getopt\(\)](#), and [my getopt\(\)](#).

Here is the call graph for this function:

8.23.3.2 my getopt()

```
int EpicsTpy::epics_list_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/l: Generates a standard listing (default) /ll: Generates a long listing /lb: Generates an autoburt save/restore file /li: Generates a LIGO DAQ ini file

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Definition at line 596 of file TpyToEpics.cpp.

References EpicsTpy::autoburt, EpicsTpy::daqini, set_listing(), set_verbose(), and EpicsTpy::standard.

Referenced by epics_list_processing(), and getopt().

Here is the call graph for this function: Here is the caller graph for this function:

8.23.3.3 operator()()

```
bool EpicsTpy::epics_list_processing::operator() (
    const ParseUtil::process_arg & arg ) [virtual]
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successfully processed

Definition at line 643 of file TpyToEpics.cpp.

References EpicsTpy::autoburt, EpicsTpy::daqini, ParseUtil::process_arg::get_alias(), EpicsTpy::split_io_support::get_file(), ParseUtil::process_arg::get_opc(), ParseUtil::substitution::get_opc(), ParseUtil::process_arg::get_process_string(), ParseUtil::process_arg::get_process_type(), EpicsTpy::split_io_support::get_processed_total(), ParseUtil::opc_list::get_properties(), ParseUtil::opc_list::get_property(), ParseUtil::substitution::has_valid_opc(), EpicsTpy::split_io_support::increment(), ParseUtil::process_arg::is_atomic(), ParseUtil::opc_list::is_published(), ParseUtil::opc_list::is_readonly(), EpicsTpy::LIGODAQ_DATATYPE_DEFAULT, EpicsTpy::LIGODAQ_DATATY \leftarrow PE_FLOAT, EpicsTpy::LIGODAQ_DATATYPE_INT32, EpicsTpy::LIGODAQ_DATATYPE_NAME, EpicsTpy::LIGODAQ_INI_HEADER, EpicsTpy::LIGODAQ_UNIT_DEFAULT, EpicsTpy::LIGODAQ_UNIT_NAME, EpicsTpy::LIGODAQ_UNIT_NONE, listing, ParseUtil::OPC_PROP_CLOSE, ParseUtil::OPC_PROP_FFST, ParseUtil::OPC_PROP_OPEN, ParseUtil::OPC_PROP_UNIT, ParseUtil::OPC_PROP_ZRST, EpicsTpy::standard, EpicsTpy::epics_conversion::to_epics(), std::trim_space(), and verbose.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.24 EpicsTpy::epics_macrofiles_processing Class Reference

Macro file processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_macrofiles_processing:

Collaboration diagram for EpicsTpy::epics_macrofiles_processing:

Public Member Functions

- [epics_macrofiles_processing \(\) noexcept=default](#)
Default constructor.
- [epics_macrofiles_processing \(macrofile_type mt\)](#)
- [epics_macrofiles_processing \(const std::stringcase &pname, const std::stringcase &dname, bool tcat3, int argc, const char *const argv\[\], bool argp\[\]=0\)](#)
- [virtual ~epics_macrofiles_processing \(\)](#)
Destructor.
- [virtual void flush \(\) noexcept](#)
flush all pending prcoossing
- [int getopt \(int argc, const char *const argv\[\], bool argp\[\]=0\) override](#)
- [int mygetopt \(int argc, const char *const argv\[\], bool argp\[\]=0\)](#)
- [virtual bool operator\(\) \(const ParseUtil::process_arg &arg\)](#)
- [macrofile_type get_macrofile_type \(\) const noexcept](#)
Get listing type.
- [void set_macrofile_type \(macrofile_type m\) noexcept](#)
Set listing.
- [void set_plcname \(const std::stringcase &name\)](#)
Set PLC name.
- [const std::stringcase & get_plcname \(\) const noexcept](#)
Get PLC name.

- bool `is_twincat3 () const noexcept`
Is this twincat 3?
- void `set_twincat3 (bool tcat3=true) noexcept`
Set twincat 3 version?
- `std::stringcase to_filename (const std::stringcase &epicsname)`
Translate epics name to filename.
- int `get_processed_total () const noexcept`
Get number of processed channels.

Static Public Attributes

- static const `std::stringcase errorstruct = "ErrorStruct"`
Type name identifying an error struct ("ErrorStruct")
- static const `std::stringcase errorlisttext2 = "_Errors.exp"`
TwinCAT 2.11: File extension identifying a list of error msgs ("_Errors.exp")
- static const `std::regex errormatchregex2`
TwinCAT 2.11: Regular expression to match the entire error record defintition.
- static const `std::stringcase errorlisttext31 = "_Errors.TcGVL"`
TwinCAT 3.1: File extension identifying a list of error msgs ("_Errors.TcGVL")
- static const `std::regex errormatchregex31`
TwinCAT 3.1: Regular expression to match the entire error record defintition.
- static const `std::regex errorsearchregex`
TwinCAT 2.11/3.1: Regular expression to search for the actual error messages.

Protected Member Functions

- bool `process_record (const macro_record &mrec, int level=0)`
Process top of stack.

Protected Attributes

- `macrofile_type macros = macrofile_type::all`
Listing type.
- `std::stringcase plcname`
PLC name.
- bool `isTwinCAT3 = false`
TwinCAT version.
- `macro_stack procstack`
Processing stack.
- int `rec_num = 0`
Current number of processed channels (records)
- `filename_set missing`
set of missing input files

8.24.1 Detailed Description

Macro file processing.

Class for generatig macro files to be used by medm

Definition at line 569 of file TpyToEpics.h.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 epics_macrofiles_processing() [1/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    macrofile_type mt )  [inline], [explicit]
```

Constructor

Parameters

<i>mt</i>	Type of macro
-----------	---------------

Definition at line 589 of file TpyToEpics.h.

8.24.2.2 epics_macrofiles_processing() [2/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    const std::stringcase & pname,
    const std::stringcase & dname,
    bool tcat3,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics_conversion::getopt](#), [multi_io_support::getopt](#) and [my getopt\(\)](#).

Parameters

<i>pname</i>	PLC name
<i>dname</i>	Directory name
<i>tcat3</i>	True if we are processing TwinCAT 3.1 files
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 758 of file TpyToEpics.cpp.

References [ParseUtil::all](#), and [my getopt\(\)](#).

Here is the call graph for this function:

8.24.3 Member Function Documentation

8.24.3.1 getopt()

```
int EpicsTpy::epics_macrofiles_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [override], [virtual]
```

Parse a command line Processed options with [epics_conversion::getopt](#) and [mygetopt\(\)](#).

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Reimplemented from [EpicsTpy::epics_conversion](#).

Definition at line 788 of file TpyToEpics.cpp.

References [EpicsTpy::epics_conversion::getopt\(\)](#), and [mygetopt\(\)](#).

Here is the call graph for this function:

8.24.3.2 mygetopt()

```
int EpicsTpy::epics_macrofiles_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/mf: Generate a macro file for each structure describing all fields /me: Generate a macro file for each structure describing the error messages /ma: Generate a macro file for each structure describing fields and errors (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Definition at line 798 of file TpyToEpics.cpp.

References EpicsTpy::all, EpicsTpy::errors, EpicsTpy::fields, and set_macrofile_type().

Referenced by epics_macrofiles_processing(), and getopt().

Here is the call graph for this function: Here is the caller graph for this function:

8.24.3.3 operator()()

```
bool EpicsTpy::epics_macrofiles_processing::operator() (
    const ParseUtil::process_arg & arg ) [virtual]
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successfully processed

Definition at line 836 of file TpyToEpics.cpp.

References EpicsTpy::macro_record::back, errorstruct, ParseUtil::process_arg::get_alias(), ParseUtil::process_arg::get_opc(), ParseUtil::substitution::get_opc(), ParseUtil::process_arg::get_process_type(), ParseUtil::process_arg::get_type_name(), ParseUtil::substitution::has_valid_opc(), ParseUtil::process_arg::is_atomic(), ParseUtil::opc_list::is_published(), ParseUtil::opc_list::is_READONLY(), EpicsTpy::macro_record::iserror, EpicsTpy::macro_info::name, process_record(), procstack, EpicsTpy::macro_info::ptype, EpicsTpy::macro_info::readonly, rec_num, EpicsTpy::macro_record::record, EpicsTpy::epics_conversion::to_epics(), and EpicsTpy::macro_info::type_n.

Here is the call graph for this function:

8.24.3.4 to_filename()

```
std::stringcase EpicsTpy::epics_macrofiles_processing::to_filename (
    const std::stringcase & epicsname )
```

Translate epics name to filename.

Translate epics name to filename

Definition at line 910 of file TpyToEpics.cpp.

References get_plcname(), and isTwinCAT3.

Referenced by process_record().

Here is the call graph for this function: Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.25 DevTc::epics_record_traits< RecType > Struct Template Reference

Epics record traits.

```
#include <devTc.h>
```

Public Types

- using `traits_type` = struct { double `val`
Epics record type.
- using `traits_type_ptr` = `epics_record_traits< RecType >::value_type *`
Epics record type pointer.
- using `value_type` = `epicsFloat64`
Value type of (raw) value field.

Static Public Member Functions

- static const char *const `name` ()
Name of the record.
- static `traits_type_ptr val` (`traits_type` *`prec`)
Returns the (raw) value of a record.
- static bool `read` (`traits_type_ptr` `epicsrec`, `plc::BaseRecord` *`baserec`)
Performs the read access on prec.
- static bool `write` (`plc::BaseRecord` *`baserec`, `traits_type_ptr` `epicsrec`)
Performs the write access on prec.

Static Public Attributes

- static const int `value_conversion` = 0
return value for read_io functions 0=default, 2=don't convert
- static const bool `input_record` = true
Indicates if this is an input record.
- static const bool `raw_record` = false
Indicates if this is a raw record.

8.25.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >
```

Epics record traits.

This traits class for Epics records.

Definition at line 304 of file devTc.h.

The documentation for this struct was generated from the following file:

- `devTc.h`

8.26 DevTc::epics_tc_db_processing Class Reference

EPICS/TCat db processing.

Inheritance diagram for DevTc::epics_tc_db_processing:

Collaboration diagram for DevTc::epics_tc_db_processing:

Public Member Functions

- `epics_tc_db_processing (TcComms::TcPLC &p, ParseUtil::replacement_rules &rules, tc_listing_def *l=nullptr, tc_macro_def *m=nullptr)`
- `~epics_tc_db_processing () override`
Destructor.
- `int getopt (int argc, const char *const argv[], bool argp[])=0 override`
- `void set_ignore_all (bool ignr) noexcept`
Set ignore for substitution list on all list.
- `bool operator() (const ParseUtil::process_arg &arg) noexcept override`
- `void flush () noexcept`
Flush output files.
- `int get_invalid_records () const noexcept`
Get number of EPICS records without tc records.

Protected Member Functions

- `epics_tc_db_processing (const epics_tc_db_processing &)=delete`
Disable copy constructor.
- `epics_tc_db_processing (epics_tc_db_processing &&)=delete`
Disable move constructor.
- `epics_tc_db_processing & operator= (const epics_tc_db_processing &)=delete`
Disable assignment operator.
- `epics_tc_db_processing & operator= (epics_tc_db_processing &&)=delete`
Disable move assignment operator.
- `void init_lists () noexcept`
Init lists.
- `void done_lists () noexcept`
Cleanup lists.
- `bool process_lists (const ParseUtil::process_arg &arg) noexcept`
- `bool process_list (filename_rule_list_tuple &listdef, const ParseUtil::process_arg &arg) noexcept`
- `void init_macros () noexcept`
Init macros.
- `void done_macros () noexcept`
Cleanup macros.
- `bool process_macros (const ParseUtil::process_arg &arg) noexcept`
- `bool process_macro (dirname_arg_macro_tuple ¯odef, const ParseUtil::process_arg &arg) noexcept`

Protected Attributes

- `TcComms::TcPLC * plc` = `nullptr`
Pointer to PLC class.
- `tc_listing_def * lists` = `nullptr`
Pointer to a set of listings.
- `tc_macro_def * macros` = `nullptr`
Pointer to macros.
- `int invnum` = 0
Number of EPICS records without tc records.

Additional Inherited Members

8.26.1 Detailed Description

EPICS/TCat db processing.

Class for generating an EPICS database and tc record

Definition at line 96 of file drvTc.cpp.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 epics_tc_db_processing()

```
DevTc::epics_tc_db_processing::epics_tc_db_processing (
    TcComms::TcPLC & p,
    ParseUtil::replacement_rules & rules,
    tc_listing_def * l = nullptr,
    tc_macro_def * m = nullptr ) [inline], [explicit]
```

Default constructor

Parameters

<code>p</code>	PLC
<code>rules</code>	Replacement rules
<code>l</code>	Pointer to list definitions
<code>m</code>	Pointer to macro definition

Definition at line 103 of file drvTc.cpp.

References `ParseUtil::replacement_rules::get_rule_table()`, and `ParseUtil::replacement_rules::is_recursive()`.

Here is the call graph for this function:

8.26.3 Member Function Documentation

8.26.3.1 getopt()

```
int DevTc::epics_tc_db_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [override], [virtual]
```

Parse a command line Processed options with epics_conversion::getopt and mygetopt.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Reimplemented from [EpicsTpy::epics_db_processing](#).

8.26.3.2 operator()()

```
bool DevTc::epics_tc_db_processing::operator() (
    const ParseUtil::process_arg & arg ) [override], [virtual], [noexcept]
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successful

Reimplemented from [EpicsTpy::epics_db_processing](#).

8.26.3.3 process_list()

```
bool DevTc::epics_tc_db_processing::process_list (
    filename_rule_list_tuple & listdef,
    const ParseUtil::process_arg & arg ) [protected], [noexcept]
```

Process a listing

Parameters

<i>listdef</i>	filename/rule pair defining a listing
<i>arg</i>	Process argument describign the variable and type

Returns

True if successful

8.26.3.4 process_lists()

```
bool DevTc::epics_tc_db_processing::process_lists (
    const ParseUtil::process_arg & arg ) [protected], [noexcept]
```

Process all listings

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successful

8.26.3.5 process_macro()

```
bool DevTc::epics_tc_db_processing::process_macro (
    dirname_arg_macro_tuple & macrodef,
    const ParseUtil::process_arg & arg ) [protected], [noexcept]
```

Process a macro

Parameters

<i>macrodef</i>	filename/rule pair defining a macro
<i>arg</i>	Process argument describign the variable and type

Returns

True if successful

8.26.3.6 process_macros()

```
bool DevTc::epics_tc_db_processing::process_macros (
    const ParseUtil::process_arg & arg ) [protected], [noexcept]
```

Process all macros

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

Returns

True if successful

The documentation for this class was generated from the following file:

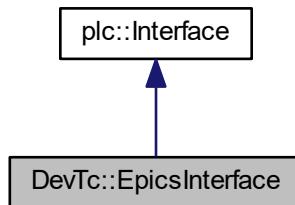
- [drvTc.cpp](#)

8.27 DevTc::EpicsInterface Class Reference

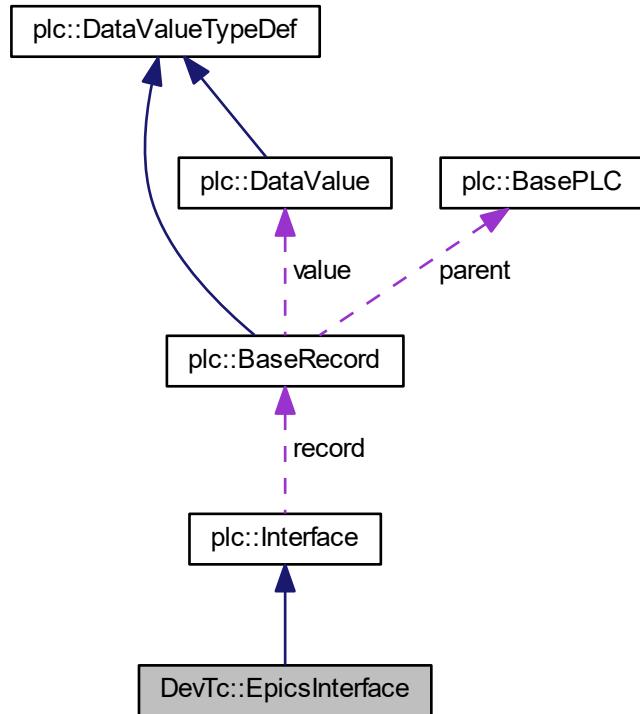
Epics interface class.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::EpicsInterface:



Collaboration diagram for DevTc::EpicsInterface:



Public Member Functions

- `EpicsInterface (plc::BaseRecord &dval) noexcept`
Constructor.
- `void set_isPassive (bool passive) noexcept`
Set isPassive.
- `bool get_isCallback () const noexcept`
Get isCallback.
- `void set_isCallback (bool isCb) noexcept`
Set isCallback.
- `void set_pEpicsRecord (dbCommon *pEpRecord) noexcept`
Set pEpicsRecord.
- `bool get_callbackRequestPending () const noexcept`
Get callbackRequestPending.
- `const epicsCallback & callback () const noexcept`
Get pointer to callback structure.
- `epicsCallback & callback () noexcept`
Get pointer to callback structure.
- `const IOSCANPVT & ioscan () const noexcept`
Get reference to io scan list pointer.
- `IOSCANPVT & ioscan () noexcept`

- *Get reference to io scan list pointer.*
- `IOSCANPVT get_ioscan () const noexcept`
 - Get pointer to io scan list.*
- `void set_ioscan (const IOSCANPVT ioscan) noexcept`
 - Set pointer to io scan list.*
- `bool push () noexcept override`
 - Makes a call to the EPICS dbProcess function.*
- `bool pull () noexcept override`
 - Does nothing.*

Static Public Member Functions

- `static int get_callback_queue_size (int pri) noexcept`
- `static int get_callback_queue_used (int pri) noexcept`
- `static int get_callback_queue_free (int pri) noexcept`
- `static int get_callback_queue_highwatermark (int pri) noexcept`
- `static int get_callback_queue_overflow (int pri) noexcept`
- `static int set_callback_queue_highwatermark_reset () noexcept`

Protected Member Functions

- `void ioscan_reset (int bitnum) noexcept`
 - Reset ioscan use flag.*

Protected Attributes

- `bool isPassive = false`
- `bool isCallback = false`
- `dbCommon * pEpicsRecord = nullptr`
 - Pointer to the EPICS record.*
- `std::mutex ioscanmux`
 - IOSCAN mutex.*
- `IOSCANPVT ioscanpvt = nullptr`
 - Pointer to IO scan list.*
- `std::atomic< unsigned int > ioscan_inuse = 0`
 - Scan in progress (bit encoded value from priorities)*
- `epicsCallback callbackval = {}`
 - Callback structure.*

Friends

- `void complete_io_scan (EpicsInterface *, IOSCANPVT, int) noexcept`

8.27.1 Detailed Description

Epics interface class.

This is a class for an EPICS Interface

Definition at line 91 of file devTc.h.

The documentation for this class was generated from the following files:

- [devTc.h](#)
- [devTc.cpp](#)

8.28 std::std::hash< std::stringcase > Struct Reference

hash for case insensitive string.

```
#include <stringcase_hash.h>
```

Public Member Functions

- std::size_t [operator\(\)](#) (const [stringcase](#) &str) const noexcept

8.28.1 Detailed Description

hash for case insensitive string.

This is a function specialization for case sensitive strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a string.

Definition at line 12 of file stringcase_hash.h.

8.28.2 Member Function Documentation

8.28.2.1 operator()()

```
std::size_t std::std::hash< std::stringcase >::operator() (
    const stringcase & str ) const [noexcept]
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv_32_str and fnv_64_str <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

Parameters

<code>str</code>	string to hash
------------------	----------------

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

- [stringcase_hash.h](#)

8.29 std::std::hash< std::wstringcase > Struct Reference

hash for case insensitive unicode string.

```
#include <stringcase_hash.h>
```

Public Member Functions

- `std::size_t operator()(const wstringcase &str) const noexcept`

8.29.1 Detailed Description

hash for case insensitive unicode string.

This is a function specialization for case sensitive unicode strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a unicode string.

Definition at line 28 of file stringcase_hash.h.

8.29.2 Member Function Documentation

8.29.2.1 operator()()

```
std::size_t std::std::hash< std::wstringcase >::operator() ( const wstringcase & str ) const [noexcept]
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv_32_str <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

Parameters

<code>str</code>	string to hash
------------------	----------------

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

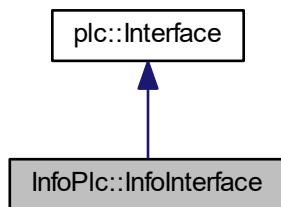
- [stringcase_hash.h](#)

8.30 InfoPlc::InfoInterface Class Reference

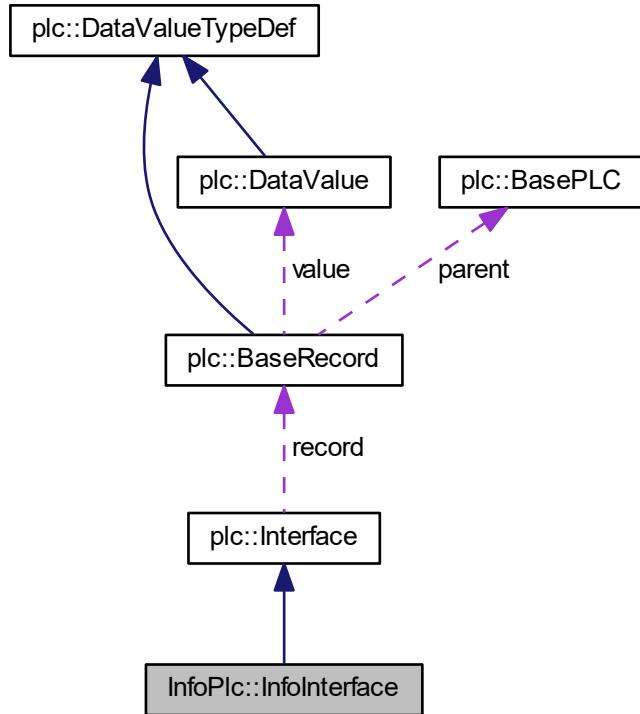
Info interface.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::InfoInterface:



Collaboration diagram for InfoPlc::InfoInterface:



Public Member Functions

- `InfoInterface (plc::BaseRecord &dval) noexcept`
- `InfoInterface (plc::BaseRecord &dval, const std::stringcase &id, const std::stringcase &name, const std::stringcase &type)`
- `bool push () noexcept override`
 push data
- `bool pull () noexcept override`
 pull data
- `virtual bool update () noexcept`
 write data
- `void printVal (FILE *fp) override`
- `const char * get_symbol_name () const noexcept override`

Static Public Member Functions

- `template<class Function>`
`static int get_infodb (const std::stringcase &prefix, const std::stringcase &plcaddr, Function &proc)`

Protected Member Functions

- bool `info_update_name () noexcept`
info update: Name of PLC
- bool `info_update_alias () noexcept`
info update: Alias name
- bool `info_update_active () noexcept`
info update: Running state of PLC
- bool `info_update_state () noexcept`
info update: AMS state of PLC
- bool `info_update_statestr () noexcept`
info update: AMS state of PLC
- bool `info_update_timestamp_str () noexcept`
info update: PLC time stamp
- bool `info_update_timestamp_local () noexcept`
info update: PLC time stamp (local time)
- bool `info_update_timestamp_year () noexcept`
info update: Year of PLC time stamp
- bool `info_update_timestamp_month () noexcept`
info update: Month of PLC time stamp
- bool `info_update_timestamp_day () noexcept`
info update: Day of PLC time stamp
- bool `info_update_timestamp_hour () noexcept`
info update: Hour of PLC time stamp
- bool `info_update_timestamp_min () noexcept`
info update: Minute of PLC time stamp
- bool `info_update_timestamp_sec () noexcept`
info update: Second of PLC time stamp
- bool `info_update_rate_read () noexcept`
info update: Period of read scanner in ms
- bool `info_update_rate_write () noexcept`
info update: Period of write scanner in ms
- bool `info_update_rate_update () noexcept`
info update: Period of update scanner in ms
- bool `info_update_records_num () noexcept`
info update: Number of EPICS records
- bool `info_update_tpy_filename () noexcept`
info update: Name of tpy file
- bool `info_update_tpy_valid () noexcept`
info update: Validity of tpy file
- bool `info_update_tpy_time_str () noexcept`
info update: Modification time of tpy file
- bool `info_update_tpy_time_year () noexcept`
info update: Year of tpy file time
- bool `info_update_tpy_time_month () noexcept`
info update: Month of tpy file time
- bool `info_update_tpy_time_day () noexcept`
info update: Day of tpy file time
- bool `info_update_tpy_time_hour () noexcept`
info update: Hour of tpy file time
- bool `info_update_tpy_time_min () noexcept`

- info update: Minute of tpy file time*
 - bool [info_update_tpy_time_sec \(\)](#) noexcept
- info update: Second of tpy file time*
 - bool [info_update_ads_version \(\)](#) noexcept
- info update: ADS library version*
 - bool [info_update_ads_revision \(\)](#) noexcept
- info update: ADS library revision*
 - bool [info_update_ads_build \(\)](#) noexcept
- info update: ADS library build*
 - bool [info_update_ads_port \(\)](#) noexcept
- info update: ADS/AMS port of PLC*
 - bool [info_update_ads_netid_str \(\)](#) noexcept
- info update: ADS/AMS address of PLC*
 - bool [info_update_ads_netid_b0 \(\)](#) noexcept
- info update: ADS/AMS address b0*
 - bool [info_update_ads_netid_b1 \(\)](#) noexcept
- info update: ADS/AMS address b1*
 - bool [info_update_ads_netid_b2 \(\)](#) noexcept
- info update: ADS/AMS address b2*
 - bool [info_update_ads_netid_b3 \(\)](#) noexcept
- info update: ADS/AMS address b3*
 - bool [info_update_ads_netid_b4 \(\)](#) noexcept
- info update: ADS/AMS address b4*
 - bool [info_update_ads_netid_b5 \(\)](#) noexcept
- info update: ADS/AMS address b5*
 - bool [info_update_svn_local \(\)](#) noexcept
- info update: SVN local modifications*
 - bool [info_update_svn_revision \(\)](#) noexcept
- info update: SVN revision, if fully committed*
 - bool [info_update_svn_time \(\)](#) noexcept
- info update: SVN compile time*
 - bool [info_update_callback_queue0_size \(\)](#) noexcept
- info update: Size of low priority callback queue*
 - bool [info_update_callback_queue0_used \(\)](#) noexcept
- info update: Used entries in low priority callback queue*
 - bool [info_update_callback_queue0_max \(\)](#) noexcept
- info update: Maximum used entries (high watermark) in low priority callback queue*
 - bool [info_update_callback_queue0_overflow \(\)](#) noexcept
- info update: Overflows in low priority callback queue*
 - bool [info_update_callback_queue0_free \(\)](#) noexcept
- info update: Free entries low priority callback queue*
 - bool [info_update_callback_queue0_percent \(\)](#) noexcept
- info update: Usage percentage of low priority callback queue*
 - bool [info_update_callback_queue0_max_prcnt \(\)](#) noexcept
- info update: Maximum percentage of low priority callback queue*
 - bool [info_update_callback_queue1_size \(\)](#) noexcept
- info update: Size of medium priority callback queue*
 - bool [info_update_callback_queue1_used \(\)](#) noexcept
- info update: Used entries in medium priority callback queue*
 - bool [info_update_callback_queue1_max \(\)](#) noexcept
- info update: Maximum used entries (high watermark) in medium priority callback queue*

- bool `info_update_callback_queue1_overflow` () noexcept
info update: Overflows in medium priority callback queue
- bool `info_update_callback_queue1_free` () noexcept
info update: Free entries medium priority callback queue
- bool `info_update_callback_queue1_percent` () noexcept
info update: Usage percentage of medium priority callback queue
- bool `info_update_callback_queue1_max_prcnt` () noexcept
info update: Maximum percentage of medium priority callback queue
- bool `info_update_callback_queue2_size` () noexcept
info update: Size of high priority callback queue
- bool `info_update_callback_queue2_used` () noexcept
info update: Used entries in high priority callback queue
- bool `info_update_callback_queue2_max` () noexcept
info update: Maximum used entries (high watermark) in high priority callback queue
- bool `info_update_callback_queue2_overflow` () noexcept
info update: Overflows in high priority callback queue
- bool `info_update_callback_queue2_free` () noexcept
info update: Free entries high priority callback queue
- bool `info_update_callback_queue2_percent` () noexcept
info update: Usage percentage of high priority callback queue
- bool `info_update_callback_queue2_max_prcnt` () noexcept
info update: Maximum percentage of high priority callback queue
- bool `info_update_callback_queue_reset_max` () noexcept
info update: reset maximum values of callback buffer queues

Protected Attributes

- `std::stringcase tCatName`
Name of TCat symbol.
- `std::stringcase tCatType`
Data type in TCat.
- `update_enum update_freq`
Update frequency.
- bool `readonly`
Readonly flag.
- `info_update_method info_update`
pointer to info update method

Static Protected Attributes

- static const `info_dbrecord_list dbinfo_list`
List of db info records.

8.30.1 Detailed Description

Info interface.

This is a class for a Info interface

Definition at line 55 of file infoPlc.h.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 InfoInterface() [1/2]

```
InfoPlc::InfoInterface::InfoInterface (
    plc::BaseRecord & dval ) [inline], [explicit], [noexcept]
```

Constructor

Parameters

<i>dval</i>	BaseRecord that this interface is part of
-------------	---

Definition at line 60 of file infoPlc.h.

8.30.2.2 InfoInterface() [2/2]

```
InfoPlc::InfoInterface::InfoInterface (
    plc::BaseRecord & dval,
    const std::string& id,
    const std::string& name,
    const std::string& type )
```

Constructor

Parameters

<i>dval</i>	BaseRecord that this interface is part of
<i>id</i>	Short name info symbol
<i>name</i>	Full name of TCat symbol
<i>type</i>	Name of TCat data type

Definition at line 639 of file infoPlc.cpp.

References dbinfo_list, InfoPlc::done, info_update, InfoPlc::once, readonly, plc::Interface::record, plc::BaseRecord::set_process(), tCatType, and update_freq.

Here is the call graph for this function:

8.30.3 Member Function Documentation

8.30.3.1 get_infodb()

```
template<class Function >
int InfoPlc::InfoInterface::get_infodb (
    const std::string& prefix,
    const std::string& plcaddr,
    Function & proc ) [static]
```

Processes all info records

Parameters

<i>prefix</i>	Prefix to channel names in the info database
<i>plcaddr</i>	PLC associated with the info record
<i>proc</i>	database processing class

Returns

Returns Number of info records processed

Definition at line 12 of file infoPlcTemplate.h.

References dbinfo_list, ParseUtil::pt_binary, ParseUtil::pt_invalid, and ParseUtil::publish.

8.30.3.2 get_symbol_name()

```
const char* InfoPlc::InfoInterface::get_symbol_name () const [inline], [override], [virtual], [noexcept]
```

Get symbol name

Returns

Symbol name associated with this interface

Reimplemented from [plc::Interface](#).

Definition at line 93 of file infoPlc.h.

References tCatName.

8.30.3.3 printVal()

```
void InfoPlc::InfoInterface::printVal (
    FILE * fp ) [override], [virtual]
```

Prints TCat symbol value and information

Parameters

<i>fp</i>	File to print symbol to
-----------	-------------------------

This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file *fp*.

Reimplemented from [plc::Interface](#).

Definition at line 703 of file infoPlc.cpp.

References [plc::BaseRecord::PlcRead\(\)](#), [plc::Interface::record](#), [tCatName](#), and [tCatType](#).

Here is the call graph for this function:

8.30.4 Member Data Documentation

8.30.4.1 dbinfo_list

```
const info\_dbrecord\_list InfoPlc::InfoInterface::dbinfo_list [static], [protected]
```

List of db info records.

List of db info tuples.

Definition at line 234 of file infoPlc.h.

Referenced by [get_infodb\(\)](#), and [InfoInterface\(\)](#).

The documentation for this class was generated from the following files:

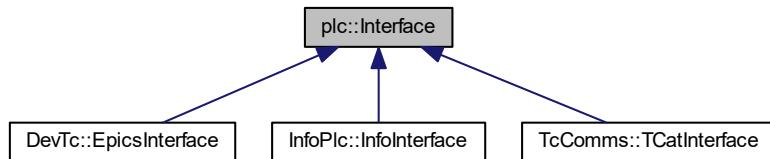
- [infoPlc.h](#)
- [infoPlc.cpp](#)
- [infoPlcTemplate.h](#)

8.31 plc::Interface Class Reference

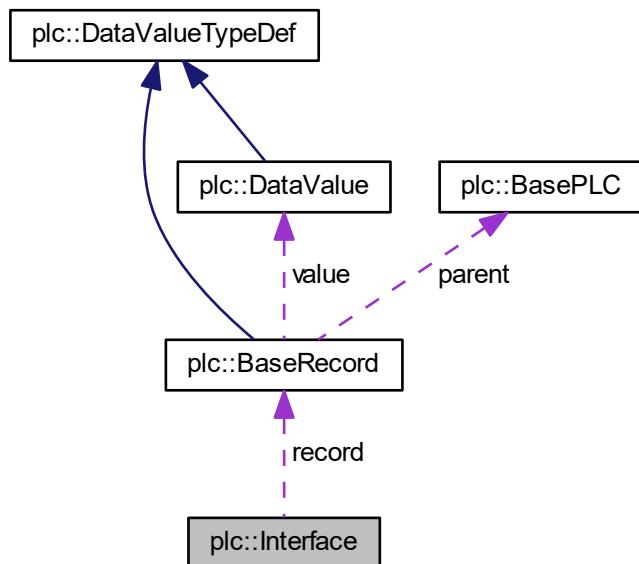
Abstract interface.

```
#include <plcBase.h>
```

Inheritance diagram for plc::Interface:



Collaboration diagram for plc::Interface:



Public Member Functions

- `Interface (BaseRecord &dval) noexcept`
 - `virtual ~Interface ()=default`
- Destructor.*
- `virtual BaseRecord & get_record () noexcept`

- `virtual const BaseRecord & get_record () const noexcept`
Return a pointer to the tag/channel record.
- `virtual bool push () noexcept=0`
Pure virtual method indicating that the value needs to be pushed.
- `virtual bool pull () noexcept=0`
Pure virtual method indicating that the value needs to be pulled.
- `BasePLC * get_parent () noexcept`
Get parent PLC that owns this record.
- `const BasePLC * get_parent () const noexcept`
Get parent PLC that owns this record.
- `virtual void printVal (FILE *fp)`
- `virtual const char * get_symbol_name () const noexcept`
Get symbol name.

Protected Attributes

- `BaseRecord & record`
Pointer to tag/channel record associated with this interface.

8.31.1 Detailed Description

Abstract interface.

This is a base class for an abstract interface to access the PLC (slave) or the user side (master).

Definition at line 33 of file plcBase.h.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 Interface()

```
plc::Interface::Interface (
    BaseRecord & dval ) [inline], [explicit], [noexcept]
```

Constructor

Parameters

<code>dval</code>	Reference to a tag/channel record
-------------------	-----------------------------------

Definition at line 38 of file plcBase.h.

8.31.3 Member Function Documentation

8.31.3.1 printVal()

```
virtual void plc::Interface::printVal (
    FILE * fp ) [inline], [virtual]
```

Print values to file

Parameters

<code>fp</code>	File pointer
-----------------	--------------

Reimplemented in [InfoPlc::InfoInterface](#), and [TcComms::TCatInterface](#).

Definition at line 58 of file plcBase.h.

Referenced by [TcComms::TcPLC::printAllRecords\(\)](#), and [TcComms::TcPLC::printRecord\(\)](#).

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

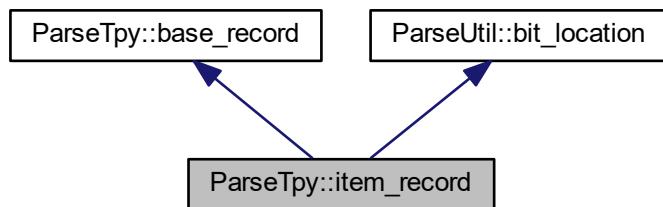
- [plcBase.h](#)
- [plcBase.cpp](#)

8.32 ParseTpy::item_record Class Reference

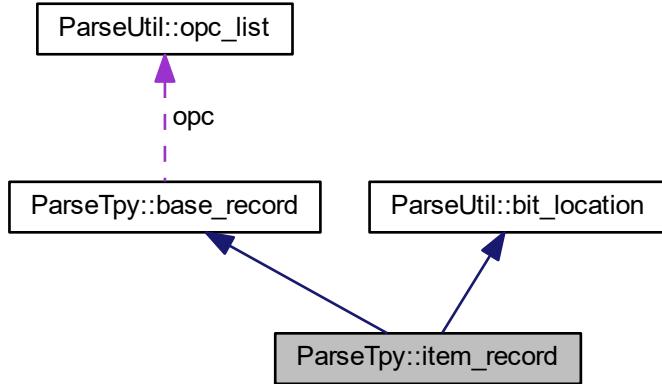
item record

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::item_record:



Collaboration diagram for ParseTpy::item_record:



Public Member Functions

- [item_record \(\) noexcept=default](#)

Default constructor.

Additional Inherited Members

8.32.1 Detailed Description

item record

This class stores typed items.

Definition at line 236 of file `ParseTpy.h`.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.33 EpicsTpy::macro_info Struct Reference

Macro information.

```
#include <TpyToEpics.h>
```

Public Member Functions

- `macro_info () noexcept=default`
Default constructor.

Public Attributes

- `ParseUtil::process_type_enum ptype = ParseUtil::process_type_enum::pt_invalid`
Process Type.
- `std::stringcase name`
name of type
- `std::stringcase type_n`
type definition
- `bool readonly = false`
readonly

8.33.1 Detailed Description

Macro information.

This structure describes a field

Definition at line 521 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

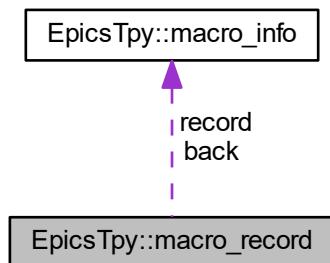
- [TpyToEpics.h](#)

8.34 EpicsTpy::macro_record Struct Reference

Macro record.

```
#include <TpyToEpics.h>
```

Collaboration diagram for EpicsTpy::macro_record:



Public Attributes

- `macro_info record`
name of structure
- `bool iserror = false`
is an ErrorStruct
- `bool haserror = false`
contains an ErrorStruct
- `int erroridx = -1`
index if fields list to ErrorStruct
- `macro_list fields`
List of fields.
- `macro_info back`
name of upper level structure

8.34.1 Detailed Description

Macro record.

This structure describes a record/struct

Definition at line 542 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

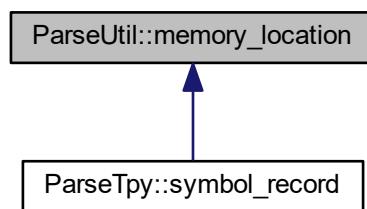
- [TpyToEpics.h](#)

8.35 ParseUtil::memory_location Class Reference

Memory location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::memory_location:



Public Member Functions

- `memory_location () noexcept=default`
Default constructor.
- `memory_location (int ig, int io, int bs) noexcept`
- `memory_location (const std::stringcase &s) noexcept`
- `bool isValid () const noexcept`
Validity.
- `int get_igroup () const noexcept`
Get IGroup.
- `void set_igroup (int ig) noexcept`
Set IGroup.
- `int get_ioffset () const noexcept`
Get IOffset.
- `void set_ioffset (int io) noexcept`
Set IOffset.
- `int get_bytesize () const noexcept`
Get BitSize.
- `void set_bytesize (int bs) noexcept`
Set BitSize.
- `bool set_section (const bit_location &loc) noexcept`
- `std::stringcase get () const`
- `bool set (const std::stringcase &s) noexcept`

Protected Attributes

- `int igrup = -1`
Memory group.
- `int ioffset = -1`
Memory offset.
- `int bytesize = -1`
Memory size in bits.

8.35.1 Detailed Description

Memory location.

This structure holds a memory location

Definition at line 288 of file ParseUtil.h.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 `memory_location()` [1/2]

```
ParseUtil::memory_location::memory_location (
    int ig,
    int io,
    int bs ) [inline], [noexcept]
```

Constructor

Parameters

<i>ig</i>	Index group
<i>io</i>	Index offset
<i>bs</i>	Size in bytes

Definition at line 297 of file ParseUtil.h.

8.35.2.2 memory_location() [2/2]

```
ParseUtil::memory_location::memory_location (
    const std::stringcase & s ) [inline], [noexcept]
```

Constructor**Parameters**

<i>s</i>	Definition string
----------	-------------------

Definition at line 301 of file ParseUtil.h.

8.35.3 Member Function Documentation**8.35.3.1 get()**

```
std::stringcase ParseUtil::memory_location::get ( ) const
```

Gets a string representation of a memory location

Returns

string with format "igroup/ioffset:size", empty on error

Definition at line 369 of file ParseUtil.cpp.

8.35.3.2 set()

```
bool ParseUtil::memory_location::set (
    const std::stringcase & s ) [noexcept]
```

Set the memory location using a string of the form: "igroup/ioffset:size" where the size is in bytes

Parameters

<code>s</code>	String describing memory location
----------------	-----------------------------------

Returns

True if successful

Definition at line 382 of file ParseUtil.cpp.

8.35.3.3 set_section()

```
bool ParseUtil::memory_location::set_section (
    const bit_location & loc ) [noexcept]
```

Set a sub section

Parameters

<code>loc</code>	Bit location within memory region
------------------	-----------------------------------

Returns

True if section within bounds, false otherwise

Definition at line 354 of file ParseUtil.cpp.

Referenced by ParseTpy::tpy_file::process_array(), and ParseTpy::tpy_file::process_type_tree().

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

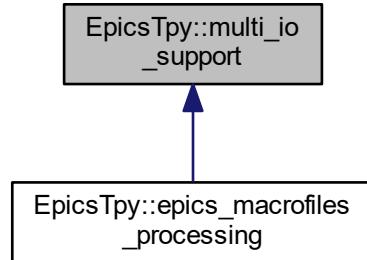
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.36 EpicsTpy::multi_io_support Class Reference

Multiple IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::multi_io_support:



Public Member Functions

- `multi_io_support ()=default`
Default constructor.
- `multi_io_support (const std::stringcase &dbname)`
Constructor.
- `multi_io_support (const std::stringcase &dbname, int argc, const char *const argv[], bool argp[])=0`
- `virtual ~multi_io_support ()`
Destructor.
- `bool operator! () const`
Return error.
- `virtual bool open (const std::stringcase &fname, const std::stringcase &io="w", bool superrmsg=false)`
Open file for reading/writing.
- `virtual void close () noexcept`
Close file.
- `FILE * get_file () const noexcept`
Get file handle.
- `void set_outdirname (const std::stringcase &dbname)`
Set output directory name.
- `const std::stringcase & get_outdirname () const noexcept`
Get output directory name.
- `void set_indirname (const std::stringcase &dbname)`
Set input directory name.
- `const std::stringcase & get_indirname () const noexcept`
Get input directory name.
- `const std::stringcase & get_filename () const noexcept`
Get full filename.
- `io_filestat fileread () const noexcept`
Reading.
- `int get_filein_total () const noexcept`
Get number of read files.
- `int get_fileout_total () const noexcept`
Get number of written files.

Protected Member Functions

- virtual int `getopt` (int argc, const char *const argv[], bool argp[] = 0)

Protected Attributes

- `std::stringcase outdirname`
Directory name.
- `std::stringcase indirname`
Directory name.
- `std::stringcase filename`
Current filename.
- `io_filestat filestat = io_filestat::closed`
reading or writing?
- `FILE * filehandle = nullptr`
Output file.
- `int file_num_in = 0`
Current file number of processed read only channels (records)
- `int file_num_out = 0`
Current file number of processed input/output channels (records)

8.36.1 Detailed Description

Multiple IO support.

Multi file IO support Supports a directory argument and opens files within

Definition at line 325 of file TpyToEpics.h.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 multi_io_support()

```
EpicsTpy::multi_io_support::multi_io_support (
    const std::stringcase & dname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

<code>dname</code>	Name of output directory
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in <code>main()</code>
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 341 of file TpyToEpics.h.

References getopt(), set_indirname(), and set_outdirname().

Here is the call graph for this function:

8.36.3 Member Function Documentation

8.36.3.1 getopt()

```
virtual int EpicsTpy::multi_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline], [protected], [virtual]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Reimplemented in [EpicsTpy::epics_macrofiles_processing](#).

Definition at line 395 of file TpyToEpics.h.

Referenced by multi_io_support().

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.37 ParseUtil::opc_list Class Reference

OPC list.

```
#include <ParseUtil.h>
```

Public Member Functions

- `opc_list () noexcept=default`
Default constructor.
- `opc_list (opc_enum state, const property_map &map)`
Constructor.
- `opc_enum get_opc_state () const noexcept`
Get opc state.
- `void set_opc_state (opc_enum state) noexcept`
Set opc state.
- `const property_map & get_properties () const noexcept`
Get opc property list.
- `property_map & get_properties () noexcept`
Get opc property list.
- `const property_map::const_iterator get_property (int key) const`
Get the specified property.
- `void add (const property_el &el)`
Add an OPC property.
- `void add (const opc_list &o)`
Add an OPC list.
- `bool is_published () const noexcept`
Is this item published?
- `bool is_READONLY () const`
Is readonly?
- `bool get_property (int prop, std::stringcase &val) const`
Get string property.
- `bool get_property (int prop, int &val) const`
Get integer property.
- `bool get_property (int prop, double &val) const`
Get real property.

Protected Attributes

- `opc_enum opc = opc_enum::no_change`
OPC state.
- `property_map opc_prop`
List of OPC properties.

8.37.1 Detailed Description

OPC list.

This class stores OPC properties.

Definition at line 165 of file ParseUtil.h.

The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.38 ParseUtil::optarg Class Reference

Optional arguments.

```
#include <ParseUtil.h>
```

Public Member Functions

- `optarg () noexcept=default`
Default constructor.
- `optarg (const std::stringcase &arg)`
- `~optarg ()`
Destructor.
- `optarg (const optarg &)=delete`
Disabled copy constructor.
- `optarg (optarg &&)=default`
Enable move constructor.
- `optarg & operator= (const optarg &)=delete`
Disabled assignment operator.
- `optarg & operator= (optarg &&)=default`
Enable move operator.
- `int parse (const std::stringcase &arg)`
- `int argc () const noexcept`
Returns the number of arguments.
- `const char *const * argv () const noexcept`
Returns the argument list.
- `const bool * argp () const noexcept`
Return the processed argument list.
- `bool * argp () noexcept`
Return the processed argument list.
- `bool all_done () const noexcept`
Returns true if there all arguments are unprocessed.

Protected Member Functions

- `void setup (size_t size) noexcept`
Setup.

Protected Attributes

- size_t **mysize** = 0
Size of allocated arrays.
- int **myargc** = 0
Number of arguments.
- std::unique_ptr<char *[]> **myargv**
Argument list.
- std::unique_ptr<bool[]> **myargp**
Processed argument list.

8.38.1 Detailed Description

Optional arguments.

This class transforms a string into a standard program argument.

Definition at line 87 of file ParseUtil.h.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 optarg()

```
ParseUtil::optarg::optarg (
    const std::string& arg) [inline], [explicit]
```

Constructor

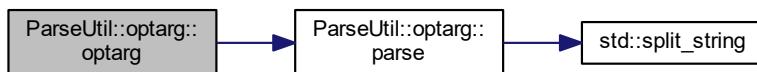
Parameters

<i>arg</i>	Option argument string
------------	------------------------

Definition at line 94 of file ParseUtil.h.

References [parse\(\)](#).

Here is the call graph for this function:



8.38.3 Member Function Documentation

8.38.3.1 parse()

```
int ParseUtil::optarg::parse (
    const std::string& arg )
```

Parse string argument

Parameters

<i>arg</i>	Option argument string
------------	------------------------

Returns

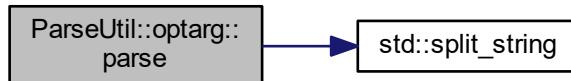
Number of processed arguments

Definition at line 170 of file ParseUtil.cpp.

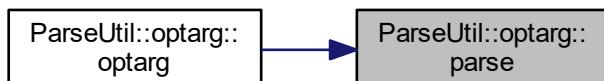
References std::split_string().

Referenced by optarg().

Here is the call graph for this function:



Here is the caller graph for this function:



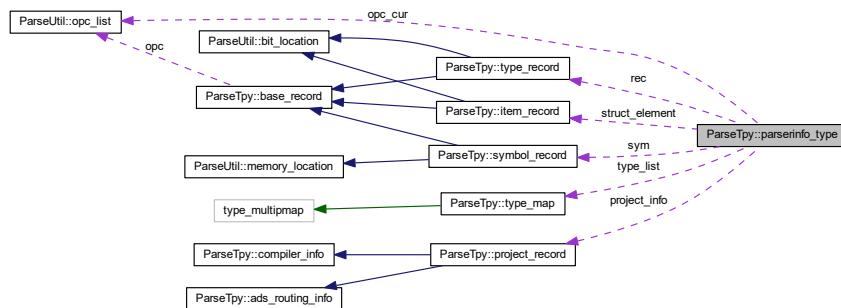
The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.39 ParseTpy::parserinfo_type Class Reference

Parser information.

Collaboration diagram for ParseTpy::parserinfo_type:



Public Member Functions

- `parserinfo_type (project_record &p, symbol_list &s, type_map &t) noexcept`
Constructor.
- `symbol_list & get_symbols () noexcept`
Get symbol list.
- `type_map & get_types () noexcept`
Get type list.
- `project_record & get_projectinfo () noexcept`
Get project information.
- `void init ()`
Initialize temporary parser info.
- `type_enum get_type_description () const noexcept`
Get type of parsed object.
- `bool verytop () noexcept`
the very top of the xml tag hierarchy (not within any tag)
- `bool top () noexcept`
the top of the xml tag hierarchy (within the PlcProjectInfo tag)

Public Attributes

- `int ignore`
ignore elements during parsing (with level)
- `bool projects`
parsing withing PlcProjectInfo tag
- `int routing`
parsing within Routing tag (1) or AdsInfo (2), Net ID (3), Port (4), Target name (5)
- `int compiler`
parsing within compiler tag (1) or compiler version (2), Twincat version (3), CPU family (4)
- `int types`
parsing within DataTypes tag (1) or DataType tag (2)

- int `symbols`
parsing within Symbols tag (1) or Symbol tag (2)
- `symbol_record sym`
temporary symbol information during parsing
- `type_record rec`
temporary type information during parsing
- int `name_parse`
- int `type_parse`
level indicator for type parsing
- `opc_list * opc_cur`
Pointer to current opc list (types only)
- `property_el opc_prop`
temporary opc element
- int `opc_parse`
level indicator for opc element
- `std::stringcase opc_data`
temporary data string for parsed opc data
- int `opc_cdata`
temporary cdata indicator
- int `igroup_parse`
level indicator for IGroup
- int `ioffset_parse`
level indicator for IOffset
- int `bitsize_parse`
level indicator for BitSize
- int `bitoffs_parse`
level indicator for BitOffs
- `std::stringcase data`
temporary data string for parsed igrup/ioffset/bitsize/bitoffs
- int `array_parse`
level indicator for array info parsing
- `std::stringcase array_data`
temporary data string for parsed array info
- `dimension array_bounds`
temporary array dimension element
- int `enum_parse`
level indicator for enum parsing
- `std::stringcase enum_data`
temporary data string for parsed enum data
- `enum_pair enum_element`
temporary enum element
- `std::stringcase enum_comment`
temporary enum comment
- int `struct_parse`
level indicator for struct parsing
- `item_record struct_element`
temporary structure element
- int `fb_parse`
level indicator for function block parsing

Protected Attributes

- `symbol_list * sym_list`
pointer to symbol list
- `type_map * type_list`
pointer to type list
- `project_record * project_info`
pointer to project info

8.39.1 Detailed Description

Parser information.

This structure keeps track of the parser information.

Definition at line 41 of file ParseTpy.cpp.

8.39.2 Member Data Documentation

8.39.2.1 name_parse

```
int ParseTpy::parserinfo_type::name_parse
```

level indicator for type name parsing level indicators in general: 0 - not encountered, 1 - parsed, 2 - currently processing, 3 - further sub tag parsing

Definition at line 97 of file ParseTpy.cpp.

The documentation for this class was generated from the following file:

- [ParseTpy.cpp](#)

8.40 ParseUtil::parserinfo_type Class Reference

Parser information.

Collaboration diagram for ParseUtil::parserinfo_type:

Public Member Functions

- `parserinfo_type (substitution_list &l) noexcept`
Constructor.
- `void init ()`
Initialze temporary parser info.
- `void add_substitution ()`
Add substitution.

Public Attributes

- int [ignore](#)
ignore elements during parsing (with level)
- int [tag](#)
CDATA indicator.
- int [cdata](#)
CDATA indicator.
- [substitution subst](#)
Substitution.
- [property_el opc_prop](#)
temporary opc element
- int [opc_parse](#)
level indicator for opc element
- [std::stringcase opc_data](#)
temporary data string for parsed opc data
- int [opc_cdata](#)
temporary cdata indicator

Protected Attributes

- [substitution_list * list](#)
pointer to symbol list

8.40.1 Detailed Description

Parser information.

This structure keeps track of the parser information.

Definition at line 43 of file ParseUtil.cpp.

8.40.2 Member Data Documentation

8.40.2.1 tag

```
int ParseUtil::parserinfo_type::tag
```

parsing withing a tag level indicators in general: 0 - not encountered, 1 - within tag, 2 - within tc 3 - within epics 4 - within properties 5 - within property

Definition at line 63 of file ParseUtil.cpp.

The documentation for this class was generated from the following file:

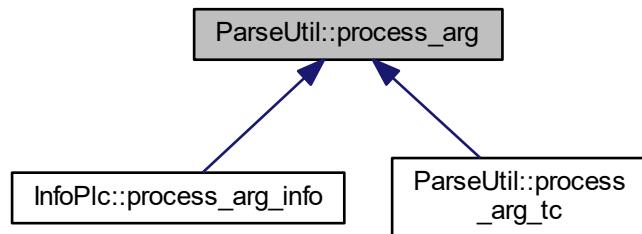
- [ParseUtil.cpp](#)

8.41 ParseUtil::process_arg Class Reference

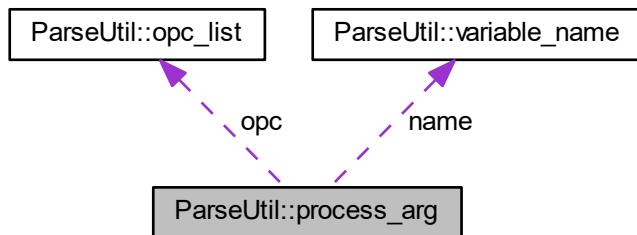
Arguments for processing.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::process_arg:



Collaboration diagram for ParseUtil::process_arg:



Public Member Functions

- `process_arg` (const `variable_name` &vname, `process_type_enum` pt, const `opc_list` &o, const `std::stringcase` &tname, bool at)
 - `virtual ~process_arg ()=default`
Destructor.
 - `process_arg` (const `process_arg` &)=default
Copy constructor.
 - `process_arg` (`process_arg` &&)=delete
Move constructor.
 - `process_arg & operator=` (const `process_arg` &)=delete
Assignment operator.

- `process_arg & operator= (process_arg &&) = delete`
Move operator.
- `const variable_name & get_var () const noexcept`
Get variable.
- `const std::stringcase & get_name () const noexcept`
Get name.
- `const std::stringcase & get_alias () const noexcept`
Get alias.
- `const std::stringcase & get_type_name () const noexcept`
Get type name.
- `const opc_list & get_opc () const noexcept`
Get OPC list.
- `process_type_enum get_process_type () const noexcept`
Get process type.
- `std::stringcase get_process_string () const`
Get process type string.
- `bool is_atomic () const noexcept`
Is atomic (or structured) type.
- `int get_size () const noexcept`
Get string length.
- `virtual std::stringcase get_full () const = 0`

Protected Member Functions

- `void deduce_size ()`
deduce string and int length from type

Protected Attributes

- `const variable_name & name`
name of type
- `const std::stringcase & type_n`
type definition
- `const opc_list & opc`
list of opc properties
- `process_type_enum ptype = process_type_enum::pt_invalid`
Process Type.
- `bool atomic = false`
Atomic element.
- `int size = 0`

8.41.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 367 of file ParseUtil.h.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 process_arg()

```
ParseUtil::process_arg::process_arg (
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

Parameters

<i>vname</i>	Variable name
<i>pt</i>	Process type
<i>o</i>	OPC list
<i>tname</i>	Type name
<i>at</i>	Atomic type

Definition at line 376 of file ParseUtil.h.

References ParseUtil::atomic.

8.41.3 Member Function Documentation

8.41.3.1 get_full()

```
virtual std::stringcase ParseUtil::process_arg::get_full ( ) const [pure virtual]
```

Gets a string representation of a PLC & memory location

Returns

string with format "prefixigroup/offset:size", empty on error

Implemented in [ParseUtil::process_arg_tc](#), and [InfoPlc::process_arg_info](#).

The documentation for this class was generated from the following files:

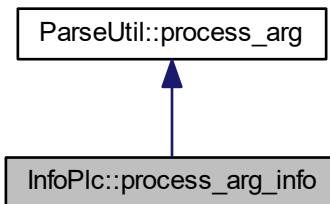
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.42 InfoPlc::process_arg_info Class Reference

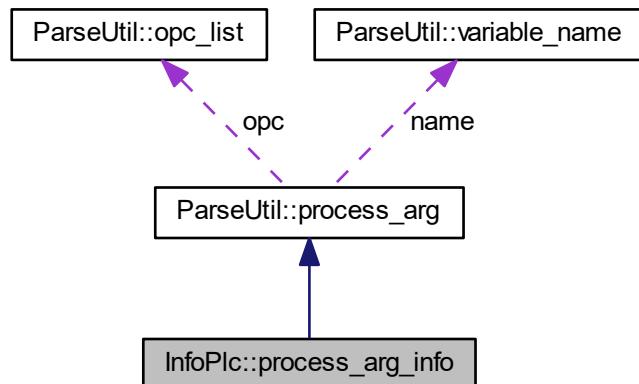
Arguments for processing.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::process_arg_info:



Collaboration diagram for InfoPlc::process_arg_info:



Public Member Functions

- `process_arg_info (const ParseUtil::variable_name &vname, ParseUtil::process_type_enum pt, const ParseUtil::opc_list &o, const std::stringcase &tnname, bool at, const std::stringcase &plcaddr)`
- `std::stringcase get_full () const override`

Protected Attributes

- const `std::stringcase & tcplc_addr`
TwinCAT PLC address.

Additional Inherited Members

8.42.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 240 of file infoPlc.h.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 process_arg_info()

```
InfoPlc::process_arg_info::process_arg_info (
    const ParseUtil::variable_name & vname,
    ParseUtil::process_type_enum pt,
    const ParseUtil::opc_list & o,
    const std::stringcase & tname,
    bool at,
    const std::stringcase & plcaddr ) [inline]
```

Constructor

Parameters

<i>vname</i>	Variable name
<i>pt</i>	Process type
<i>o</i>	OPC list
<i>tname</i>	Type name
<i>at</i>	Atomic type
<i>plcaddr</i>	TwinCAT PLC address

Definition at line 250 of file infoPlc.h.

8.42.3 Member Function Documentation

8.42.3.1 get_full()

```
std::stringcase InfoPlc::process_arg_info::get_full () const [override], [virtual]
```

Gets a string representation of a PLC & memory location

Returns

string with format "prefixigroup/offset:size", empty on error

Implements [ParseUtil::process_arg](#).

Definition at line 1533 of file infoPlc.cpp.

The documentation for this class was generated from the following files:

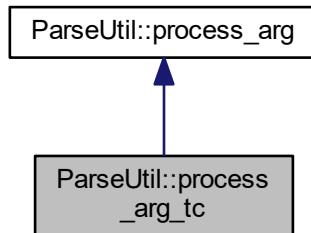
- [infoPlc.h](#)
- [infoPlc.cpp](#)

8.43 ParseUtil::process_arg_tc Class Reference

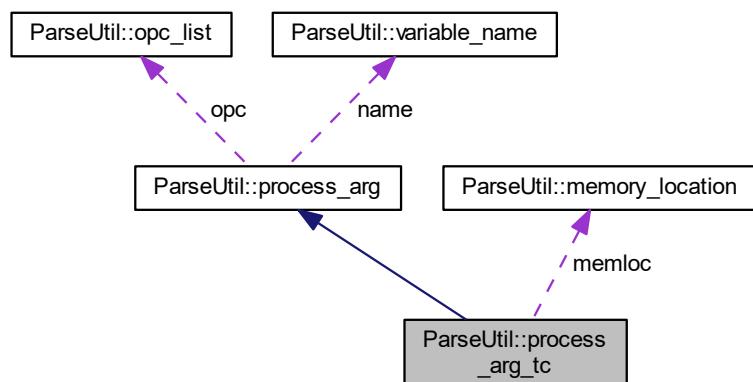
Arguments for processing.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::process_arg_tc:



Collaboration diagram for ParseUtil::process_arg_tc:



Public Member Functions

- `process_arg_tc` (const `memory_location` &loc, const `variable_name` &vname, `process_type_enum` pt, const `opc_list` &o, const `std::stringcase` &tname, bool at)
- int `get_igroup` () const noexcept
Get IGroup.
- int `get_ioffset` () const noexcept
Get IOffset.
- int `get_bytesize` () const noexcept
Get BitSize.
- `std::stringcase` `get` () const
- `std::stringcase` `get_full` () const override

Protected Attributes

- const `memory_location` & memloc
memory location

Additional Inherited Members

8.43.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 436 of file ParseUtil.h.

8.43.2 Constructor & Destructor Documentation

8.43.2.1 `process_arg_tc()`

```
ParseUtil::process_arg_tc::process_arg_tc (
    const memory_location & loc,
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

Parameters

<code>loc</code>	Memory location
<code>vname</code>	Variable name
<code>pt</code>	Process type
<code>o</code>	OPC list
<code>tname</code>	Type name
<code>at</code>	Atomic type

Definition at line 446 of file ParseUtil.h.

8.43.3 Member Function Documentation

8.43.3.1 get()

```
std::stringcase ParseUtil::process_arg_tc::get () const [inline]
```

Gets a string representation of a memory location

Returns

string with format "igroup/ioffset:size", empty on error

Definition at line 459 of file ParseUtil.h.

8.43.3.2 get_full()

```
std::stringcase ParseUtil::process_arg_tc::get_full () const [override], [virtual]
```

Gets a string representation of a PLC & memory location

Returns

string with format "prefixigroup/ioffset:size", empty on error

Implements [ParseUtil::process_arg](#).

Definition at line 453 of file ParseUtil.cpp.

References [ParseUtil::OPC_PROP_PLNAME](#).

The documentation for this class was generated from the following files:

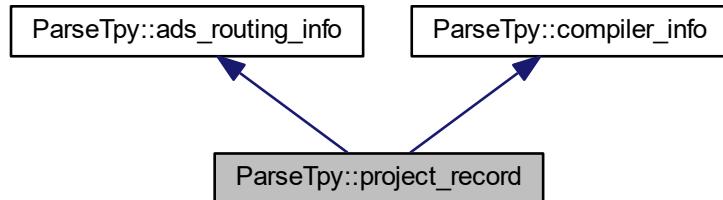
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.44 ParseTpy::project_record Class Reference

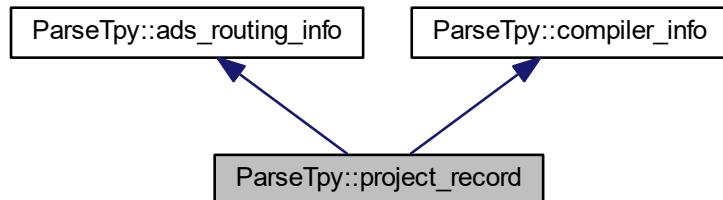
Project information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::project_record:



Collaboration diagram for ParseTpy::project_record:



Public Member Functions

- [project_record \(\)=default](#)
Default constructor.

Additional Inherited Members

8.44.1 Detailed Description

Project information.

This is a base class for storing the project information

Definition at line 136 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.45 DevTc::register_devsup Class Reference

Device support registration.

```
#include <devTc.h>
```

Collaboration diagram for DevTc::register_devsup:

DevTc::register_devsup ↗ the_register_devsup

Public Types

- using `link_func` = `bool(&)(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Type describing the link function.
- using `test_pattern` = `std::pair< std::regex, link_func & >`
pair of pattern and link function
- using `test_pattern_list` = `std::vector< test_pattern >`
list of pattern/link functions

Static Public Member Functions

- static void `add` (const std::regex &rgx, `link_func` &func) noexcept
Register a pattern/link function.
- static bool `linkRecord` (const std::stringcase &inpout, dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord) noexcept
linkRecord

Protected Member Functions

- `register_devsup ()` noexcept
Default constructor (adds linkTcRecord entry)
- `register_devsup (const register_devsup &)=delete`
Disabled copy constructor.
- `register_devsup (register_devsup &&)=delete`
Disabled move constructor.
- `register_devsup & operator= (const register_devsup &)=delete`
Disabled assignment operator.
- `register_devsup & operator= (register_devsup &&)=delete`
Disabled move assignment operator.

Protected Attributes

- `test_pattern_list tp_list`

list of pattern and links

Static Protected Attributes

- static `register_devsup the_register_devsup`

the one global instance of the register class

8.45.1 Detailed Description

Device support registration.

This is a class for managing device support for multiple record types, such as TwinCAT/ADS and Info.

Definition at line 42 of file devTc.h.

The documentation for this class was generated from the following files:

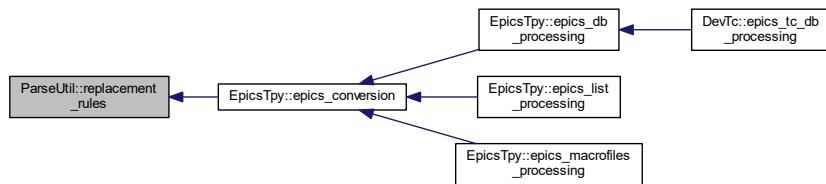
- `devTc.h`
- `devTc.cpp`

8.46 ParseUtil::replacement_rules Class Reference

Replacement rules.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::replacement_rules:



Public Member Functions

- `replacement_rules (bool rec=true) noexcept`
Default constructor.
- `replacement_rules (const replacement_table &t, bool rec=true)`
Constructor.
- `replacement_rules (const std::stringcase &s, const std::stringcase alias="")`
- `void clear ()`
Clear.
- `void add_rule (const std::stringcase &var, const std::stringcase &val)`
Add a rule.
- `void set_rule_table (const replacement_table &t)`
set table
- `replacement_table & get_rule_table () noexcept`
get table
- `const replacement_table & get_rule_table () const noexcept`
get table
- `std::stringcase apply_replacement_rules (const std::stringcase &s) const`
replace
- `bool HasRules () const noexcept`
Has rules.
- `bool is_recursive () const noexcept`
Is recursive?
- `void set_recursive (bool rec) noexcept`
Set recursive.
- `bool parse_rules (const std::stringcase &s, const std::stringcase alias="")`
- `std::stringcase get_rule_string () const`
Get the string representing the rules.

Static Public Attributes

- `static const char *const prefix = "${"`
prefix for replacement rule: \${
- `static const char *const suffix = "}"`
suffix for replacement rule: }

Protected Attributes

- `replacement_table table`
Replacement table.
- `bool recursive = true`
Recursive replacement.

8.46.1 Detailed Description

Replacement rules.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 27 of file ParseUtil.h.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 replacement_rules()

```
ParseUtil::replacement_rules::replacement_rules (
    const std::string& s,
    const std::string& alias = "") [inline], [explicit]
```

Constructor

Parameters

<i>s</i>	Rule string of the form VAR1=VAL1,VAR2=VAL2,...
<i>alias</i>	Alias name

Definition at line 37 of file ParseUtil.h.

References parse_rules().

Here is the call graph for this function:

8.46.3 Member Function Documentation

8.46.3.1 parse_rules()

```
bool ParseUtil::replacement_rules::parse_rules (
    const std::string& s,
    const std::string& alias = "")
```

Parse a rule string

Parameters

<i>s</i>	Rule string of the form VAR1=VAL1,VAR2=VAL2,...
<i>alias</i>	Alias name

Returns

True if proper rule string

Definition at line 93 of file ParseUtil.cpp.

References ParseUtil::OPC_NAME_ALIAS, and std::trim_space().

Referenced by replacement_rules().

Here is the call graph for this function: Here is the caller graph for this function:

The documentation for this class was generated from the following files:

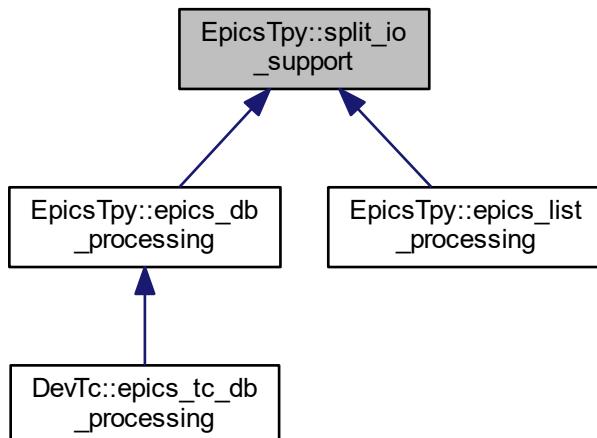
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.47 EpicsTpy::split_io_support Class Reference

Split IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::split_io_support:



Public Member Functions

- [split_io_support \(\) noexcept=default](#)
Default constructor.
- [split_io_support \(const std::string& fname, bool split=false, int max=0\)](#)
Constructor.
- [split_io_support \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [virtual ~split_io_support \(\)](#)
Destructor.
- [split_io_support \(const split_io_support &\)](#)
- [split_io_support \(split_io_support &&\) = delete](#)
Move constructor.
- [split_io_support & operator= \(const split_io_support &\)](#)

- `split_io_support & operator= (split_io_support &&)=delete`
Move assignment.
- `virtual bool operator! () const noexcept`
Return error.
- `bool increment (bool readonly)`
- `virtual void flush () noexcept`
Flush contents of output files.
- `FILE * get_file () const noexcept`
Get output file.
- `const std::stringcase & get_filename () const noexcept`
Get output filename.
- `bool is_split () const noexcept`
Is output split?
- `int get_max () const noexcept`
Maximum of channels per file.
- `int get_processed_total () const noexcept`
Get number of processed channels.
- `int get_processed_READONLY () const noexcept`
Get number of processed readonly channels.
- `int get_processed_io () const noexcept`
Get number of processed input/output channels.

Protected Member Functions

- `virtual void set_filename (const std::stringcase &fname)`
Set output filename.
- `virtual void close () noexcept`
Close files.
- `void set_split (bool split) noexcept`
set split
- `void set_max (int max) noexcept`
Set maximum of channels per file.
- `virtual int getopt (int argc, const char *const argv[], bool argp[] = 0)`

Protected Attributes

- `bool error = true`
Error.
- `std::stringcase outfilename`
Output filename.
- `bool split_io = false`
Split output into read only channels and input/output channels.
- `int split_n = 0`
Maximum number of channels per file; 0 indicates no limit.
- `FILE * outf = stdout`
Output file.
- `FILE * outf_in = nullptr`
Output file for read only channels.
- `FILE * outf_io = nullptr`

- int `rec_num` = 0
Current number of processed channels (records)
- int `rec_num_in` = 0
Current number of processed read only channels (records)
- int `rec_num_io` = 0
Current number of processed input/output channels (records)
- int `file_num_in` = 1
Current file number of processed read only channels (records)
- int `file_num_io` = 1
Current file number of processed input/output channels (records)
- `std::stringcase file_num_in_s`
Contains the readonly file number in string format.
- `std::stringcase file_num_io_s`
Contains the input/output file number in string format.
- `std::stringcase file_in_s`
Contains the file extenstion for readonly files.
- `std::stringcase file_io_s`
Contains the file extenstion for input/output files.

8.47.1 Detailed Description

Split IO support.

Split file IO support Output can be split in multiple files if the number of channels exceeds the maximum specified for a file

Definition at line 179 of file TpyToEpics.h.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 split_io_support() [1/2]

```
EpicsTpy::split_io_support::split_io_support (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

<code>fname</code>	Filename for output
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 197 of file TpyToEpics.h.

References getopt(), and set_filename().

Here is the call graph for this function:

8.47.2.2 split_io_support() [2/2]

```
EpicsTpy::split_io_support::split_io_support (
    const split_io_support & iosup )
```

Copy constructor File pointers will be moved over and the original ones will become invalid.

Definition at line 252 of file TpyToEpics.cpp.

8.47.3 Member Function Documentation

8.47.3.1 getopt()

```
int EpicsTpy::split_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [protected], [virtual]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ysio: Splits database into input only and input/output records /nsio: Does not split database by record type (default) /sn 'num': Splits database or listing into files with no more than num records /sn 0: Does not split database or listing into multiple files (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

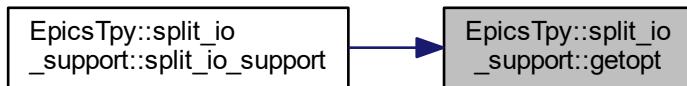
Reimplemented in [EpicsTpy::epics_list_processing](#), [DevTc::epics_tc_db_processing](#), and [EpicsTpy::epics_db_processing](#).

Definition at line 290 of file TpyToEpics.cpp.

References set_max(), and set_split().

Referenced by split_io_support().

Here is the call graph for this function: Here is the caller graph for this function:



8.47.3.2 increment()

```
bool EpicsTpy::split_io_support::increment (
    bool readonly )
```

Increase the channel number

Parameters

<i>readonly</i>	Indicates if channel is readonly
-----------------	----------------------------------

Returns

True if no error

Definition at line 417 of file TpyToEpics.cpp.

References error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Referenced by EpicsTpy::epics_list_processing::operator()().

Here is the caller graph for this function:

8.47.3.3 operator=(*)

```
split_io_support & EpicsTpy::split_io_support::operator= (
    const split_io_support & iosup )
```

Copy assignment File pointers will be moved over and the original ones will become invalid.

Definition at line 263 of file TpyToEpics.cpp.

References close(), error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.48 ParseUtil::substitution Class Reference

Channel substitution.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::substitution:

Collaboration diagram for ParseUtil::substitution:

Public Member Functions

- [substitution \(\) noexcept=default](#)
Default constructor.
- [substitution \(const std::stringcase &n\)](#)
- [substitution \(const char *s\)](#)
- [substitution \(const std::stringcase &n, const std::stringcase &a\)](#)
- [bool has_valid_opc \(\) const noexcept](#)
Is there a valid OPC list?
- [void set_opc_valid \(bool valid=true\) noexcept](#)
Sets valid flag for OPC list.
- [const ParseUtil::opc_list & get_opc \(\) const noexcept](#)
Get OPC list.
- [ParseUtil::opc_list & get_opc \(\) noexcept](#)
Get OPC list.
- [bool get_used \(\) const noexcept](#)
Is there a valid OPC list?
- [void set_used \(bool uis=true\) const noexcept](#)
Sets valid flag for OPC list.

Protected Attributes

- [bool valid_opc = false](#)
Contains a valid OPC list.
- [opc_list opc](#)
OPC list.
- [bool used_in_substitution = false](#)
used in substitution

8.48.1 Detailed Description

Channel substitution.

Class to specify a channel substitution

Definition at line 562 of file ParseUtil.h.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 substitution() [1/3]

```
ParseUtil::substitution::substitution (
    const std::stringcase & n ) [inline], [explicit]
```

Constructor

Parameters

<code>n</code>	Name and alias
----------------	----------------

Definition at line 569 of file ParseUtil.h.

8.48.2.2 substitution() [2/3]

```
ParseUtil::substitution::substitution (
    const char * s ) [inline], [explicit]
```

Constructor

Parameters

<code>s</code>	Name and alias
----------------	----------------

Definition at line 573 of file ParseUtil.h.

8.48.2.3 substitution() [3/3]

```
ParseUtil::substitution::substitution (
    const std::stringcase & n,
    const std::stringcase & a ) [inline]
```

Constructor

Parameters

<i>n</i>	Name
<i>a</i>	Alias

Definition at line 578 of file ParseUtil.h.

The documentation for this class was generated from the following file:

- [ParseUtil.h](#)

8.49 ParseUtil::substitution_list Class Reference

List of channel substitutions.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::substitution_list:

Public Member Functions

- **substitution_list ()=default**
Default constructor.
- **substitution_list (process_substitution_enum process) noexcept**
Constructor.
- **substitution_list (FILE *inp, process_substitution_enum process=process_substitution_enum::standard) noexcept**
Constructor.
- **bool parse (FILE *inp) noexcept**
Parse a file.
- **bool parse (const char *p, int len) noexcept**
Parse a memory region.
- **void add_substitution (const substitution &subst)**
Add a replacement name.
- **bool query_substitution (std::stringcase &name, bool published, const substitution **subst=nullptr) const**
- **const substitution_map & get_substitution_list () const noexcept**
Get substitution list.
- **void set_substitution_list (const substitution_map &l)**
Set substitution list.
- **bool get_ignore () const noexcept**
Get ignore.
- **void set_ignore (bool ignr) noexcept**
Set ignore.
- **process_substitution_enum get_processing () const noexcept**
Get ignore_notfound.
- **void set_processing (process_substitution_enum process) noexcept**
Set ignore_notfound.
- **bool get_try_no_array_index () const noexcept**
Get try_no_array_index.
- **void set_try_no_array_index (bool nai) noexcept**
Set try_no_array_index.
- **void check_unused_substitutions () const noexcept**

Protected Member Functions

- void `parse_finish () noexcept`
called after finish parsing

Protected Attributes

- `substitution_map list`
Replacement table.
- bool `ignore = false`
Ignore list all together.
- `process_substitution_enum processing = process_substitution_enum::standard`
Ignore if not found, otherwise use same name.
- bool `try_no_array_index = true`
Try lookup with no array indices when direct name look up fails.

8.49.1 Detailed Description

List of channel substitutions.

Class to specify a list of channel substitutions

Definition at line 624 of file ParseUtil.h.

8.49.2 Member Function Documentation

8.49.2.1 query_substitution()

```
bool ParseUtil::substitution_list::query_substitution (
    std::stringcase & name,
    bool published,
    const substitution ** subst = nullptr ) const
```

Query for replacement name Returns true, if a valid name is returned, false otherwise

Parameters

<code>name</code>	Lookup name, will be replaced if found
<code>published</code>	This variable is published by default
<code>subst</code>	Pointer to substitution, if found (null otherwise)

Definition at line 614 of file ParseUtil.cpp.

References ParseUtil::variable_name::get_alias(), ParseUtil::ignore, and ParseUtil::substitution::set_used().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

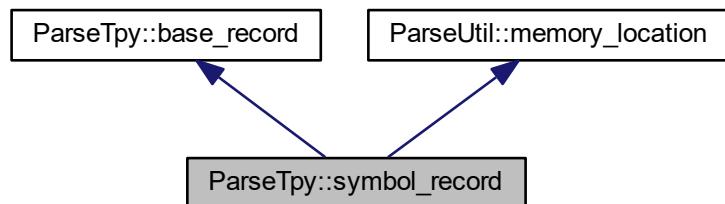
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.50 ParseTpy::symbol_record Class Reference

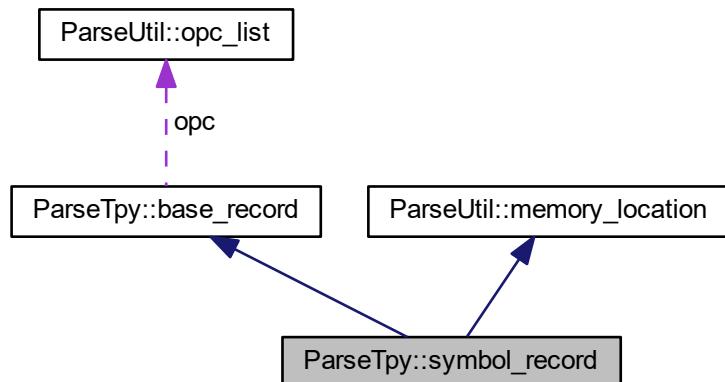
Symbol record.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::symbol_record:



Collaboration diagram for ParseTpy::symbol_record:



Public Member Functions

- [symbol_record \(\)=default](#)
Default constructor.
- [const memory_location & get_location \(\) const noexcept](#)
get memory location
- [memory_location & get_location \(\) noexcept](#)
get memory location

Additional Inherited Members

8.50.1 Detailed Description

Symbol record.

This structure holds a symbol record

Definition at line 339 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.51 syminfo_processing Class Reference

Symbol processing.

Public Member Functions

- [syminfo_processing \(FILE *outfile=0, bool atomic=true\) noexcept](#)
Constructor.
- [bool operator\(\) \(const process_arg &arg\) const noexcept](#)
Process.

Protected Attributes

- [FILE * outf](#)
Ouptut file.
- [bool firstline](#)
Firstline?

8.51.1 Detailed Description

Symbol processing.

Symbol processing

Definition at line 16 of file ParseTpyInfo.cpp.

The documentation for this class was generated from the following file:

- [ParseTpyInfo.cpp](#)

8.52 plc::System Class Reference

[System](#) to keep track of PLCs.

```
#include <plcBase.h>
```

Public Types

- using `mutex_type` = std::recursive_mutex
Defines the mutex type.
- using `guard` = std::lock_guard< `mutex_type` >
Defined the mutex guard type.

Public Member Functions

- bool `add` (BasePLC *plc)
- bool `add` (BasePLCPtr plc)
- `BasePLCPtr find` (std::stringcase id)
Finds a PLC by its name.
- template<typename func >
`void for_each` (func &f)
- template<typename func >
`void for_each` (const func &f)
- `void printVals` ()
Print all PLC record values to the console.
- `void printVal` (const std::string &var)
- `void start` () noexcept
Start scanning after ioc is running.
- `void stop` () noexcept
Stop scanning when ioc is paused.
- bool `is_ioc_running` () const noexcept
get loc run state
- `void set_ioc_state` (bool run) noexcept
set loc run state

Static Public Member Functions

- static [System & get \(\) noexcept](#)
Return a reference to the global System variable.

Protected Attributes

- [mutex_type mux](#)
Mutex to synchronize access to this class.
- [BasePLCList PLCs](#)
Master list of all PLCs.
- [bool locRun](#)
IOC is running.

8.52.1 Detailed Description

[System](#) to keep track of PLCs.

This is a class for managing multiple PLCs.

Definition at line 899 of file plcBase.h.

8.52.2 Member Function Documentation

8.52.2.1 add() [1/2]

```
bool plc::System::add (
    BasePLC * plc )  [inline]
```

Add a new PLC, PLC will be adopted

Parameters

<i>plc</i>	Pointer to plc
------------	----------------

Returns

true if successful

Definition at line 912 of file plcBase.h.

8.52.2.2 add() [2/2]

```
bool plc::System::add (
    BasePLCPtr plc )
```

Add a new PLC

Parameters

<i>plc</i>	Smart pointer to plc
------------	----------------------

Returns

true if successful

Definition at line 770 of file plcBase.cpp.

References mux, and PLCs.

8.52.2.3 for_each() [1/2]

```
template<typename func >
void plc::System::for_each (
    const func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

<i>f</i>	Function which takes BaseRecord* as the argument
----------	--

Definition at line 252 of file plcBaseTemplate.h.

References mux, and PLCs.

8.52.2.4 for_each() [2/2]

```
template<typename func >
void plc::System::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

<i>f</i>	Function which takes BaseRecord* as the argument
----------	--

Definition at line 241 of file plcBaseTemplate.h.

References mux, and PLCs.

Referenced by start(), and stop().

Here is the caller graph for this function:

8.52.2.5 printVal()

```
void plc::System::printVal (
    const std::string & var )
```

Print PLC record value to the console

Parameters

<i>var</i>	Variable name or wildcard
------------	---------------------------

Definition at line 795 of file plcBase.cpp.

References mux, and PLCs.

Referenced by DevTc::tcPrintVal().

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

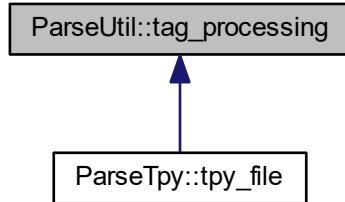
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.53 ParseUtil::tag_processing Class Reference

Tag processing selection.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::tag_processing:



Public Member Functions

- `tag_processing () noexcept=default`
Default constructor.
- `tag_processing (bool all, process_tag_enum proctags, bool nostring=false) noexcept`
- `tag_processing (int argc, const char *const argv[], bool argp[] = 0) noexcept`
- `int getopt (int argc, const char *const argv[], bool argp[] = 0) noexcept`
- `bool get_export_all () const noexcept`
Get the export rule.
- `void set_export_all (bool all) noexcept`
Set the export rule.
- `process_tag_enum get_process_tags () const noexcept`
Get the process rule.
- `void set_process_tags (process_tag_enum proctags) noexcept`
Set the process rule.
- `bool get_no_strings () const noexcept`
Get the string rule.
- `void set_no_strings (bool nostring) noexcept`
Set the string rule.

Protected Attributes

- `bool export_all = false`
Process all symbols regardless of opc publish setting.
- `process_tag_enum process_tags = process_tag_enum::all`
Process only atomic types.
- `bool no_string_tags = false`
Don't process strings.

8.53.1 Detailed Description

Tag processing selection.

Class to specify which symbols and tags/names to process

Definition at line 485 of file ParseUtil.h.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 tag_processing() [1/2]

```
ParseUtil::tag_processing::tag_processing (
    bool all,
    process_tag_enum proctags,
    bool nostring = false ) [inline], [noexcept]
```

Constructor

Parameters

<i>all</i>	Process all tags
<i>proctags</i>	Process atomic and/or structured tags
<i>nostring</i>	Don't process string tags

Definition at line 494 of file ParseUtil.h.

8.53.2.2 tag_processing() [2/2]

```
ParseUtil::tag_processing::tag_processing (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline], [noexcept]
```

Constructor Commaline arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 504 of file ParseUtil.h.

References [ParseUtil::optarg::argc\(\)](#), [ParseUtil::optarg::argp\(\)](#), and [ParseUtil::optarg::argv\(\)](#).

Here is the call graph for this function:

8.53.3 Member Function Documentation

8.53.3.1 getopt()

```
int ParseUtil::tag_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } ) [noexcept]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ea: Export all variables regardless of OPC setting /eo: Only export variables which are marked as OPC export (default) /ns: No string variables are processed /ys: String variables are processes (default) /pa: Call process for all types (default) /ps: Call process for simple (atomic) types only /pc: Call process for complex (structure and array) types only

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the contructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Definition at line 469 of file ParseUtil.cpp.

Referenced by [main\(\)](#).

Here is the caller graph for this function:

The documentation for this class was generated from the following files:

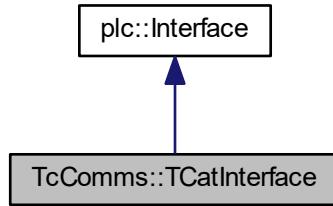
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.54 TcComms::TCatInterface Class Reference

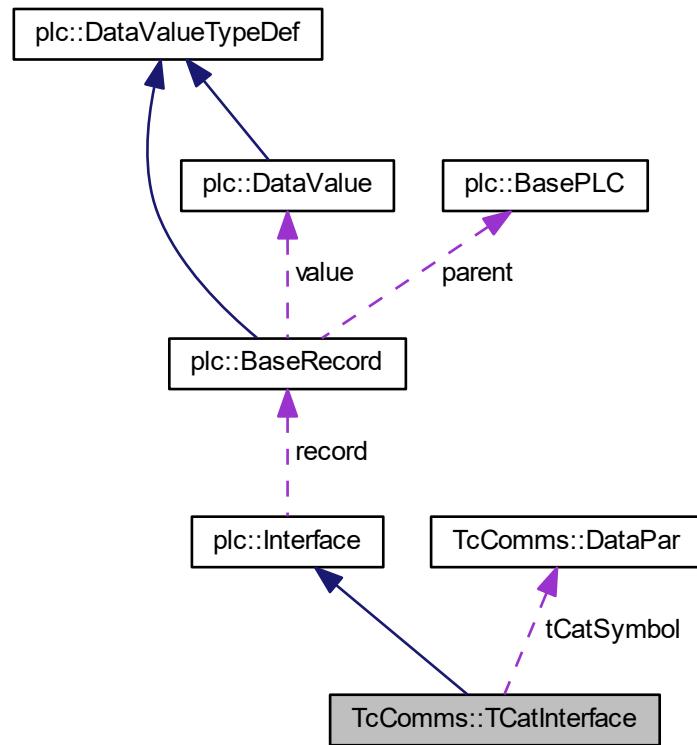
TCat interface class.

```
#include <tcComms.h>
```

Inheritance diagram for TcComms::TCatInterface:



Collaboration diagram for TcComms::TCatInterface:



Public Member Functions

- `TCatInterface (plc::BaseRecord &dval) noexcept`
Constructor.

- `TCatInterface (plc::BaseRecord &dval, const std::stringcase &name, unsigned long group, unsigned long offset, unsigned long length, const std::stringcase &ttype, bool isStruct, bool isEnum)`
- `const std::stringcase get_tCateName () const noexcept`
Get name of TCat symbol.
- `void set_tCateName (std::stringcase name) noexcept`
Set name of TCat symbol.
- `const char * get_symbol_name () const noexcept override`
Get symbol name.
- `const std::stringcase get_tCateType () noexcept`
Get TCat data type.
- `void set_tCateType (std::stringcase type) noexcept`
Set TCat data type.
- `DataPar get_tCateSymbol () noexcept`
Get structure containing index group, index offset, size.
- `unsigned long get_indexGroup () const noexcept`
Get index group.
- `void set_indexGroup (unsigned long group) noexcept`
Set index group.
- `unsigned long get_indexOffset () const noexcept`
Get index offset.
- `void set_indexOffset (unsigned long offset) noexcept`
Set index offset.
- `unsigned long get_size () const noexcept`
Get size in bytes of symbol.
- `void set_size (unsigned long nBytes) noexcept`
Set size in bytes of symbol.
- `size_t get_requestOffs () const noexcept`
Get offset into response buffer.
- `void set_requestOffs (size_t pVal) noexcept`
Set offset into response buffer.
- `virtual TcPLC * get_parent () noexcept`
Get parent PLC that owns this record.
- `virtual const TcPLC * get_parent () const noexcept`
Get parent PLC that owns this record.
- `int get_requestNum () noexcept`
Get the request group number this record is in.
- `void set_requestNum (int rNum) noexcept`
Set the request group number this record is in.
- `void printVal (FILE *fp) noexcept override`
- `bool push () noexcept override`
Does nothing.
- `bool pull () noexcept override`
Does nothing.

Protected Attributes

- `std::stringcase tCatName`
Name of TCat symbol.
- `std::stringcase tCatType`
Data type in TCat.
- `DataPar tCatSymbol`
Struct storing index group, index offset, and length of TC symbol.
- `int requestNum`
Which request group in the PLC.
- `size_t requestOffs`
Offset into response buffer.

8.54.1 Detailed Description

TCat interface class.

This is a class for a TCat interface

Definition at line 70 of file tcComms.h.

8.54.2 Constructor & Destructor Documentation

8.54.2.1 TCatInterface()

```
TcComms::TCatInterface::TCatInterface (
    plc::BaseRecord & dval,
    const std::stringcase & name,
    unsigned long group,
    unsigned long offset,
    unsigned long length,
    const std::stringcase & type,
    bool isStruct,
    bool isEnum )
```

Constructor

Parameters

<code>dval</code>	BaseRecord that this interface is part of
<code>name</code>	Name of TCat symbol
<code>group</code>	Index group of TCat symbol
<code>offset</code>	Index offset of TCat symbol
<code>length</code>	Size in bytes of TCat symbol
<code>type</code>	Name of TCat data type
<code>isStruct</code>	True = this symbol is a structure in TCat
<code>isEnum</code>	True = this symbol is an enum in TCat

Definition at line 88 of file tcComms.cpp.

8.54.3 Member Function Documentation

8.54.3.1 printVal()

```
void TcComms::TCatInterface::printVal (
    FILE * fp ) [override], [virtual], [noexcept]
```

Prints TCat symbol value and information

Parameters

<i>fp</i>	File to print symbol to
-----------	-------------------------

This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file fp.

Reimplemented from [plc::Interface](#).

Definition at line 132 of file tcComms.cpp.

References TcComms::TcPLC::get_responseBuffer().

Here is the call graph for this function:

The documentation for this class was generated from the following files:

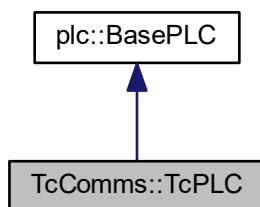
- [tcComms.h](#)
- [tcComms.cpp](#)

8.55 TcComms::TcPLC Class Reference

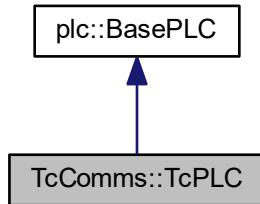
TwinCAT PLC.

```
#include <tcComms.h>
```

Inheritance diagram for TcComms::TcPLC:



Collaboration diagram for TcComms::TcPLC:



Public Types

- using `buffer_type` = `char`
Buffer type.
- using `buffer_ptr` = `std::shared_ptr< buffer_type >`
Smart pointer to buffer.

Public Member Functions

- `TcPLC (std::string tpyPath)`
Constructor.
- `~TcPLC () override`
Destructor.
- `bool is_valid_tpy () noexcept`
Is typ still valid? Meaning, it hasn't changed.
- `AmsAddr get_addr () const noexcept`
Get AMS netID of TwinCAT system and port number for this PLC.
- `bool set_addr (std::stringcase netid, int port)`
- `long get_nReadPort () const noexcept`
Get read port number.
- `long get_nWritePort () const noexcept`
Get write port number.
- `int get_read_scanner_multiple () const noexcept`
Get slowdown multiple for EPICS read.
- `void set_read_scanner_multiple (int mult) noexcept`
Set slowdown multiple for EPICS read.
- `ADSSTATE get_ads_state () const noexcept`
Get ADS state.
- `bool is_read_active () const noexcept`
Is read scanner active and successful.
- `const std::string & get_tpyfilename () const noexcept`
Get the tpy filename.
- `bool is_tpyfile_valid () const noexcept`
Is the tpy file valid?

- `time_t get_tpyfile_time () const noexcept`
Get the file modification time of the tpy file.
- `bool start () override`
Starts the appropriate scanners.
- `bool optimizeRequests ()`
- `buffer_ptr get_responseBuffer (size_t idx) noexcept`
- `void printAllRecords () override`
Prints symbol information for entire list of symbols to console.
- `void printRecord (const std::string &var) override`

Protected Member Functions

- `void read_scanner () override`
Makes read requests to ADS, makes PlcWrite on all data values.
- `void write_scanner () override`
Collects records to be written to TCat, makes write request.
- `void update_scanner () override`
Makes sure we don't have stale values.
- `void set_ads_state (ADSSTATE state) noexcept`
Set ADS state.
- `void setup_ads_notification () noexcept`
Set up ADS status change notification.
- `void remove_ads_notification () noexcept`
Remove ADS status change notification.
- `long openPort () noexcept`
Opens a new ADS communication port.
- `void closePort (long nPort) noexcept`

Protected Attributes

- `std::mutex sync`
Mutex.
- `AmsAddr addr`
AMS netID of TwinCAT system and port number for this PLC.
- `std::string pathTpy`
The path of the tpy file.
- `time_t timeTpy`
Modification time of file.
- `std::atomic< bool > checkTpy`
need to check modification time to make sure tpy file hasn't changed
- `bool validTpy`
tpy file is valid and hasn't changed
- `int nRequest`
Number of read request groups.
- `std::vector< DataPar > adsGroupReadRequestVector`
Vector of index group, index offset, size for read requests.
- `std::vector< buffer_ptr > adsResponseBufferVector`
Vector of buffers for each read request group.
- `plc::BaseRecordList nonTcRecords`

- List of all records that don't interface directly with a PLC (info)*
- int `scanRateMultiple`
Slowdown multiple for EPICS read.
 - int `cyclesLeft`
 - int `update_workload`
Workload for update scanner.
 - `plc::BaseRecordPtr update_last`
last updated record
 - `std::atomic<ADSSTATE> ads_state`
ADS state.
 - unsigned long `ads_handle`
ADS handle.
 - `std::atomic<bool> ads_restart`
ADS restart.
 - long `nReadPort`
Port number for ADS read connection.
 - long `nWritePort`
Port number for ADS write connection.
 - long `nNotificationPort`
Port number for ADS notification connection.
 - bool `read_active`
read active and successful

Friends

- void __stdcall `ADScallback` (AmsAddr *, AdsNotificationHeader *, unsigned long)
Notification callback is a friend.

8.55.1 Detailed Description

TwinCAT PLC.

Class for a connection to a TwinCAT PLC This class is derived from a BasePLC object, and specializes in managing records that contain a plc interface for TCat. This class is initialized using a .tpy file, from which it will obtain the AMS address information for connecting with TCat through ADS.

Reading and writing from/to ADS will be managed by this class, with read requests being grouped by continuous memory region in TCat to optimize read scanning for speed. Write requests are made using an ADS sum request.

There is also an option to send a request to ADS to check the status of both the PLC device and also the ADS connection.

Definition at line 255 of file tcComms.h.

8.55.2 Member Function Documentation

8.55.2.1 `closePort()`

```
void TcComms::TcPLC::closePort (
    long nPort ) [protected], [noexcept]
```

Closes an ADS communication port

Parameters

<i>nPort</i>	Number of port to close
--------------	-------------------------

Definition at line 894 of file tcComms.cpp.

Referenced by remove_ads_notification().

Here is the caller graph for this function:

8.55.2.2 get_responseBuffer()

```
TcPLC::buffer_ptr TcComms::TcPLC::get_responseBuffer (
    size_t idx ) [noexcept]
```

Get pointer to the beginning of a read request response buffer

Parameters

<i>idx</i>	Index of response buffer
------------	--------------------------

Returns

pointer to buffer

Definition at line 579 of file tcComms.cpp.

Referenced by TcComms::TCatInterface::printVal().

Here is the caller graph for this function:

8.55.2.3 optimizeRequests()

```
bool TcComms::TcPLC::optimizeRequests ( )
```

Sorts read channels into request groups. Will make a new request group for channels not in continuous memory region in TCat. Will create buffers of appropriate size for each read request, and let each TCat record know where in the read response buffer the data for that symbol is.

Returns

true if successful

Definition at line 462 of file tcComms.cpp.

8.55.2.4 printRecord()

```
void TcComms::TcPLC::printRecord (
    const std::string & var ) [override], [virtual]
```

Print a record values to stdout. (override for action)

Parameters

<code>var</code>	variable name (accepts wildcards)
------------------	-----------------------------------

Reimplemented from [plc::BasePLC](#).

Definition at line 655 of file tcComms.cpp.

References `plc::Interface::get_symbol_name()`, `plc::Interface::printVal()`, and `plc::BasePLC::records`.

Here is the call graph for this function:

8.55.2.5 `set_addr()`

```
bool TcComms::TcPLC::set_addr (
    std::stringcase netid,
    int port )
```

Set AMS address

Returns

true if successful

Definition at line 365 of file tcComms.cpp.

8.55.2.6 `set_ads_state()`

```
void TcComms::TcPLC::set_ads_state (
    ADSSTATE state ) [protected], [noexcept]
```

Set ADS state.

[TcPLC::set_ads_state](#)

Definition at line 704 of file tcComms.cpp.

Referenced by `TcComms::ADSCallback()`, and `setup_ads_notification()`.

Here is the caller graph for this function:

8.55.2.7 `update_scanner()`

```
void TcComms::TcPLC::update_scanner () [override], [protected], [virtual]
```

Makes sure we don't have stale values.

what!?

Reimplemented from [plc::BasePLC](#).

Definition at line 853 of file tcComms.cpp.

References `ads_restart`, `get_ads_state()`, `plc::BasePLC::get_next()`, `is_read_active()`, `plc::BasePLC::mux`, `plc::BasePLC::name`, `plc::BasePLC::records`, `remove_ads_notification()`, `setup_ads_notification()`, `update_last`, and `update_workload`.

Here is the call graph for this function:

8.55.3 Friends And Related Function Documentation

8.55.3.1 ADScallback

```
void __stdcall ADScallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId ) [friend]
```

Notification callback is a friend.

Callback for ADS state change

Definition at line 683 of file tcComms.cpp.

Referenced by setup_ads_notification().

8.55.4 Member Data Documentation

8.55.4.1 cyclesLeft

```
int TcComms::TcPLC::cyclesLeft [protected]
```

Cycles until EPICS read will be made Counts down from scanRateMultiple, resets at 0

Definition at line 376 of file tcComms.h.

Referenced by read_scanner().

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

8.56 TcComms::tcProcWrite Class Reference

TwinCAT process write requests.

```
#include <tcComms.h>
```

Public Member Functions

- `tcProcWrite` (const AmsAddr &a, long amsport, size_t mrec=1000) noexcept
Default constructor.
- `~tcProcWrite ()`
Destructor: will process the TCat writes.
- `tcProcWrite (tcProcWrite &&tp)` noexcept
Move constructor.
- `tcProcWrite & operator=(tcProcWrite &&)` noexcept
Move assignment operator.
- `void operator() (plc::BaseRecord *prec)`
Process on record.
- `void * read_ptr (int sz)` noexcept
- `bool add (long igrp, long ioffs, long sz)`

Protected Member Functions

- `bool check_alloc (int extra=0)` noexcept
Checks if we have enough memory allocated.
- `void tcwrite ()` noexcept
writes the current header/data to TCat

Protected Attributes

- `AmsAddr addr`
AMS address.
- `long port`
Port to be used to write to TCat.
- `char * ptr`
Pointer to header to be written.
- `char * data`
Pointer to data to be written.
- `size_t maxrec`
Maximum number of individual requests.
- `size_t size`
Size of data.
- `size_t alloc`
Size of allocated header/data array.
- `size_t count`
Current number of individual requests.
- `std::vector<tcProcWrite> req`

8.56.1 Detailed Description

TwinCAT process write requests.

Class for collecting and processing write requests This class iterates through the entire record list on the PLC and collects those records whose data value has a dirty flag set on the plc side. These records are then sent as a group to ADS.

In order to not overload the ADS server, a maximum number of symbols per request is defined, and should not be > 2000.

Definition at line 175 of file tcComms.h.

8.56.2 Member Function Documentation

8.56.2.1 add()

```
bool TcComms::tcProcWrite::add (
    long igrup,
    long ioffs,
    long sz )
```

Add header info

Parameters

<i>igrup</i>	iGroup number for tc write
<i>ioffs</i>	iOffset number for tc write
<i>sz</i>	Size of data to be written

Returns

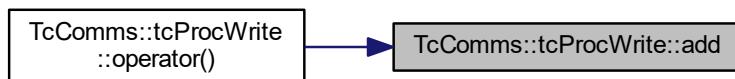
true if successful

Definition at line 251 of file tcComms.cpp.

References check_alloc(), count, maxrec, ptr, and req.

Referenced by operator()().

Here is the call graph for this function: Here is the caller graph for this function:



8.56.2.2 read_ptr()

```
void * TcComms::tcProcWrite::read_ptr (
    int sz ) [noexcept]
```

Get a pointer to read the value in

Parameters

sz	Requested size
----	----------------

Definition at line 239 of file tcComms.cpp.

Referenced by operator()().

Here is the caller graph for this function:

8.56.3 Member Data Documentation

8.56.3.1 req

```
std::vector<tcProcWrite> TcComms::tcProcWrite::req [protected]
```

Queued TCat requests (each would have reached maxrec individual requests)

Definition at line 224 of file tcComms.h.

Referenced by add(), and ~tcProcWrite().

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

8.57 DevTc::tcRegisterTolocShell Class Reference

Register TC commands.

```
#include <drvTc.h>
```

8.57.1 Detailed Description

Register TC commands.

Register TC commands to IOC shell This class registers the callback functions for the TC IOC commands

Definition at line 18 of file drvTc.h.

The documentation for this class was generated from the following files:

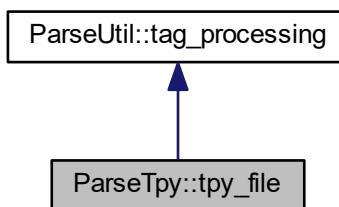
- [drvTc.h](#)
- [drvTc.cpp](#)

8.58 ParseTpy::tpy_file Class Reference

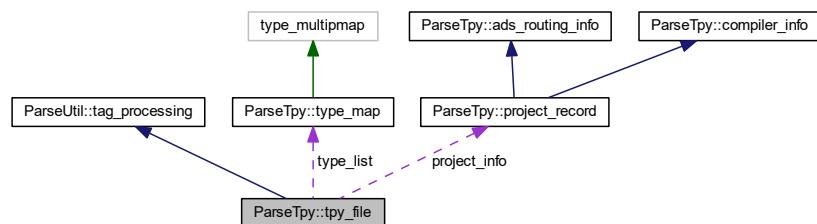
Tpy file parsing.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::tpy_file:



Collaboration diagram for ParseTpy::tpy_file:



Public Member Functions

- [tpy_file \(\)=default](#)
Default constructor.
- [tpy_file \(FILE *inp\)](#)
Constructor.
- [bool parse \(FILE *inp\)](#)
Parse a file.
- [bool parse \(const char *p, int len\)](#)
Parse a memory region.
- [const symbol_list & get_symbols \(\) const noexcept](#)
Return list of symbols.
- [const type_map & get_types \(\) const noexcept](#)
Return list of types.
- [const project_record & get_project_info \(\) const noexcept](#)

- Return project information.*
- template<class Function>


```
int process_symbols (Function &process, const std::stringcase &prefix=std::stringcase()) const
    Process the type tree of a symbol.
```
 - template<class Function>


```
int process_type_tree (const symbol_record &symbol, Function &process, const std::stringcase &prefix=std::stringcase()) const
    Process the type tree of a symbol.
```
 - template<class Function>


```
int process_type_tree (const type_record &typ, ParseUtil::opc_list defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const
    Process the type tree of a type.
```
 - template<class Function>


```
int process_type_tree (const std::stringcase &typ, unsigned int id, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const
    Process the type tree of a type.
```

Protected Member Functions

- void `parse_finish ()`

finish up the parsing
- template<class Function>


```
int process_array (const ParseTpy::type_record &typ, ParseTpy::dimensions dim, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname, int level) const
    Process the type tree of a type.
```

Protected Attributes

- `project_record project_info`

Project information.
- `symbol_list sym_list`

List of symbols.
- `type_map type_list`

List of types.

8.58.1 Detailed Description

Tpy file parsing.

This class holds the structure of a tpy file

Definition at line 359 of file ParseTpy.h.

8.58.2 Member Function Documentation

8.58.2.1 parse_finish()

```
void ParseTpy::tpy_file::parse_finish ( ) [protected]
```

finish up the parsing

This function is called at the end of parsing. Here we set the TC server name in the OPC variables for each symbol

Definition at line 506 of file ParseTpy.cpp.

References ParseUtil::OPC_PROP_PLCNAME.

8.58.2.2 process_array()

```
template<class Function >
int ParseTpy::tpy_file::process_array (
    const ParseTpy::type_record & typ,
    ParseTpy::dimensions dim,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname,
    int level ) const [protected]
```

Process the type tree of a type.

Resolves the type information for an array. Calls the process function for each index with an argument of type process_arg.

Parameters

<i>typ</i>	Name of type to resolve
<i>dim</i>	Dimensions of the array
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

Returns

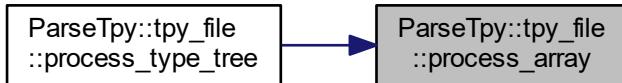
Number of processes variables

Definition at line 219 of file ParseTpyTemplate.h.

References ParseUtil::all, ParseUtil::variable_name::append(), ParseUtil::bit_location::get_bit_size(), ParseTpy::base_record::get_name(), ParseUtil::variable_name::get_name(), ParseUtil::tag_processing::get_process_tags(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), process_type_tree(), ParseUtil::pt_binary, ParseUtil::bit_location::set_bit_size(), ParseUtil::memory_location::set_section(), and ParseUtil::structured.

Referenced by process_type_tree().

Here is the call graph for this function: Here is the caller graph for this function:



8.58.2.3 process_symbols()

```
template<class Function >
int ParseTpy::tpy_file::process_symbols (
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Iterates over the symbol list and processes all specified tags.

Parameters

<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

Returns

Number of processes variables

Definition at line 9 of file ParseTpyTemplate.h.

References ParseUtil::tag_processing::get_export_all(), get_symbols(), process_type_tree(), and ParseUtil::publish.

Referenced by main().

Here is the call graph for this function: Here is the caller graph for this function:



8.58.2.4 process_type_tree() [1/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const std::stringcase & typ,
    unsigned int id,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
    int level = 0 ) const
```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like STRING) is found. Then, calls the process function with an argument of type process_arg.

Parameters

<i>typ</i>	Name of type to resolve
<i>id</i>	Decoration or unique ID of type
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

Returns

Number of processes variables

Definition at line 164 of file ParseTpyTemplate.h.

References ParseUtil::all, ParseUtil::atomic, ParseTpy::type_map::find(), ParseUtil::variable_name::get_name(), ParseUtil::tag_processing::get_no_strings(), ParseUtil::tag_processing::get_process_tags(), process_type_tree(), ParseUtil::pt_bool, ParseUtil::pt_int, ParseUtil::pt_real, ParseUtil::pt_string, and type_list.

Here is the call graph for this function:

8.58.2.5 process_type_tree() [2/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const symbol_record & symbol,
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Starts with a symbol and resolves the type information recursively until an atomic type (like INT) is found. Then, calls the process function with an argument of type process_arg. The function must return true if successful and false otherwise.

Parameters

<i>symbol</i>	Symbol to resolve
<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

Returns

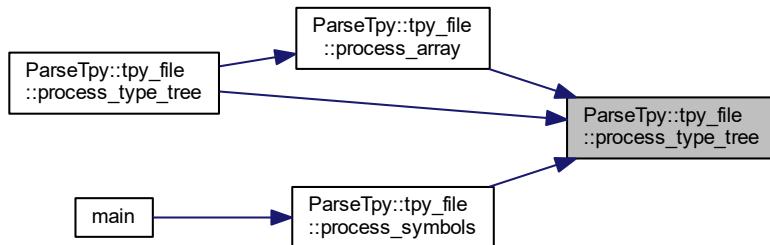
Number of processes variables

Definition at line 25 of file ParseTpyTemplate.h.

References ParseUtil::variable_name::append(), ParseUtil::memory_location::get_bytesize(), ParseTpy::base_record::get_name(), ParseUtil::variable_name::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), and ParseTpy::base_record::get_type_pointer().

Referenced by process_array(), process_symbols(), and process_type_tree().

Here is the call graph for this function: Here is the caller graph for this function:

**8.58.2.6 process_type_tree() [3/3]**

```

template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const type_record & typ,
    ParseUtil::opc_list defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
    int level = 0 ) const

```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like BOOL) is found. Then, calls the process function with an argument of type process_arg.

Parameters

<i>typ</i>	Type to resolve
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

Returns

Number of processes variables

Definition at line 56 of file ParseTpyTemplate.h.

References ParseUtil::opc_list::add(), ParseUtil::all, ParseUtil::variable_name::append(), ParseTpy::arraytype, ParseUtil::atomic, ParseTpy::enumtype, ParseTpy::functionblock, ParseTpy::type_record::get_array_dimensions(), ParseTpy::type_record::get_enum_list(), ParseTpy::base_record::get_name(), ParseUtil::variable_name::get_name(), ParseTpy::base_record::get_opc(), ParseUtil::tag_processing::get_process_tags(), ParseUtil::opc_list::get_properties(), ParseTpy::type_record::get_struct_list(), ParseTpy::base_record::get_type_decoration(), ParseTpy::type_record::get_type_description(), ParseTpy::base_record::get_type_name(), process_array(), process_type_tree(), ParseUtil::pt_binary, ParseUtil::pt_enum, ParseUtil::pt_int, ParseUtil::memory_location::set_section(), ParseTpy::simple, ParseTpy::structtype, and ParseUtil::structured.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

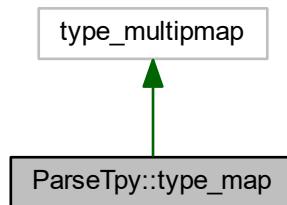
- [ParseTpy.h](#)
- [ParseTpy.cpp](#)
- [ParseTpyTemplate.h](#)

8.59 ParseTpy::type_map Class Reference

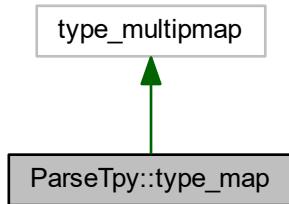
Type dictionary.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_map:



Collaboration diagram for ParseTpy::type_map:



Public Member Functions

- [type_map \(\)=default](#)
Constructor.
- [const value_type::second_type * find \(value_type::first_type id, const std::stringcase &typn\) const](#)
find an element
- [int patch_type_decorators \(\)](#)

8.59.1 Detailed Description

Type dictionary.

This is a map of type records, index is type number as defined in tpy

Definition at line 319 of file ParseTpy.h.

The documentation for this class was generated from the following files:

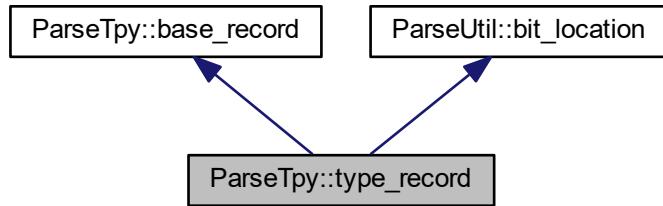
- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.60 ParseTpy::type_record Class Reference

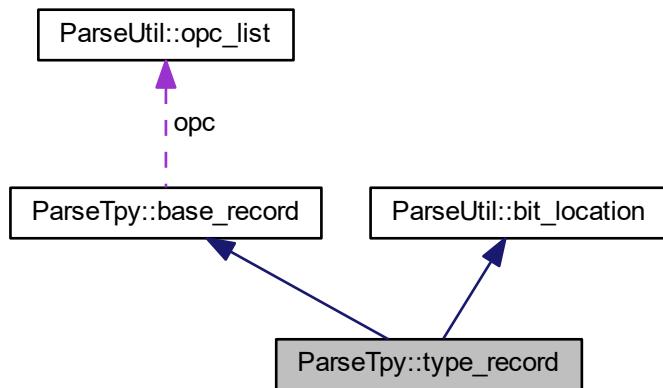
Type record information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_record:



Collaboration diagram for ParseTpy::type_record:



Public Member Functions

- `type_record () noexcept=default`
Default constructor.
- `type_enum get_type_description () const noexcept`
get the data type
- `void set_type_description (type_enum desc) noexcept`
Set decoration (type ID)
- `unsigned int get_name_decoration () const noexcept`
Get decoration (type ID)
- `void set_name_decoration (unsigned int id) noexcept`
Set decoration (type ID)
- `const dimensions & get_array_dimensions () const noexcept`
Get array dimensions.

- [dimensions & get_array_dimensions \(\) noexcept](#)
Get array dimensions.
- [const enum_map & get_enum_list \(\) const noexcept](#)
Get enumerated list.
- [enum_map & get_enum_list \(\) noexcept](#)
Get enumerated list.
- [const item_list & get_struct_list \(\) const noexcept](#)
Get structure list.
- [item_list & get_struct_list \(\) noexcept](#)
Get structure list.

Protected Attributes

- [type_enum type_desc = type_enum::unknown](#)
Type description.
- [unsigned int name_decoration = 0](#)
decoration (type ID) of type name
- [dimensions array_list](#)
table of dimensions
- [enum_map enum_list](#)
map of enum id and name
- [item_list struct_subitems](#)
list of structure elements

8.60.1 Detailed Description

Type record information.

This structure holds a type record

Definition at line 270 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.61 ParseUtil::variable_name Class Reference

Variable name.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::variable_name:

Public Member Functions

- `variable_name () noexcept=default`
Default constructor.
- `variable_name (const std::stringcase &n)`
Constructor.
- `variable_name (const char *s)`
Constructor.
- `variable_name (const std::stringcase &n, const std::stringcase &a)`
Constructor.
- `const std::stringcase & get_name () const noexcept`
Get name.
- `std::stringcase & get_name () noexcept`
Get name.
- `const std::stringcase & get_alias () const noexcept`
Get alias.
- `std::stringcase & get_alias () noexcept`
Get alias.
- `void set (const std::stringcase &n)`
Set name & alias.
- `void set (const std::stringcase &n, const std::stringcase &a)`
Set name & alias.
- `void append (const std::stringcase &n, const std::stringcase &sep=".")`
Append.
- `void append (const std::stringcase &n, const opc_list &opc, const std::stringcase &sep=".")`
Append.

Protected Attributes

- `std::stringcase name`
variable name
- `std::stringcase alias`
alias name

8.61.1 Detailed Description

Variable name.

This is a class for storing a variable name and an alias

Definition at line 213 of file ParseUtil.h.

The documentation for this class was generated from the following files:

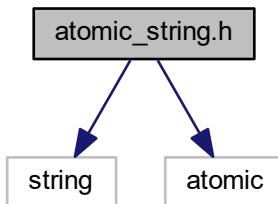
- `ParseUtil.h`
- `ParseUtil.cpp`

Chapter 9

File Documentation

9.1 atomic_string.h File Reference

```
#include <string>
#include <atomic>
Include dependency graph for atomic_string.h:
```



This graph shows which files directly or indirectly include this file:

Classes

- class `std::atomic_string< stringT >`
atomic strings
- class `std::atomic< string >`
atomic<string>
- class `std::atomic< wstring >`
atomic<wstring>

9.1.1 Detailed Description

Header which includes a class for implementing an atomic specialization for string.

9.2 devTc.cpp File Reference

```
#include "devTc.h"
#include "ParseTpy.h"
#include "infoPlc.h"
#include "errlog.h"
#include "dbAccess.h"
#include "dbCommon.h"
#include "dbEvent.h"
#include "recSup.h"
#include "epicsExport.h"
#include "epicsRingPointer.h"
#include <iostream>
Include dependency graph for devTc.cpp:
```

Namespaces

- [DevTc](#)
Namespace for TCat device support.

Functions

- void [DevTc::complete_io_scan](#) ([EpicsInterface](#) *epics, IOSCANPVT ioscan, int prio) noexcept
- int [DevTc::get_callback_queue_size](#) (int pri)
- int [DevTc::get_callback_queue_used](#) (int pri)
- int [DevTc::get_callback_queue_free](#) (int pri)
- int [DevTc::get_callback_queue_highwatermark](#) (int pri)
- int [DevTc::get_callback_queue_overflow](#) (int pri)
- int [DevTc::set_callback_queue_highwatermark_reset](#) (void)
- [epicsExportAddress](#) (dset, aival_record_tc_dset)
Record processing entry for ai.
- [epicsExportAddress](#) (dset, airval_record_tc_dset)
Record processing entry for raw ai.
- [epicsExportAddress](#) (dset, aaival_record_tc_dset)
Record processing entry for aai.
- [epicsExportAddress](#) (dset, bival_record_tc_dset)
Record processing entry for bi.
- [epicsExportAddress](#) (dset, birval_record_tc_dset)
Record processing entry for raw bi.
- [epicsExportAddress](#) (dset, longinval_record_tc_dset)
Record processing entry for longin.
- [epicsExportAddress](#) (dset, mbbival_record_tc_dset)
Record processing entry for mbbi.
- [epicsExportAddress](#) (dset, mbbirval_record_tc_dset)
Record processing entry for raw mbbi.
- [epicsExportAddress](#) (dset, mbbiDirectval_record_tc_dset)
Record processing entry for mbbiDirect.
- [epicsExportAddress](#) (dset, mbbiDirectrval_record_tc_dset)
Record processing entry for raw mbbiDirect.
- [epicsExportAddress](#) (dset, stringinval_record_tc_dset)
Record processing entry for stringin.

- **epicsExportAddress** (dset, lsival_record_tc_dset)
Record processing entry for lsi (long string)
- **epicsExportAddress** (dset, waveformval_record_tc_dset)
Record processing entry for waveform.
- **epicsExportAddress** (dset, aoval_record_tc_dset)
Record processing entry for ao.
- **epicsExportAddress** (dset, aorval_record_tc_dset)
Record processing entry for raw ao.
- **epicsExportAddress** (dset, aaoval_record_tc_dset)
Record processing entry for aao.
- **epicsExportAddress** (dset, boaval_record_tc_dset)
Record processing entry for bo.
- **epicsExportAddress** (dset, borval_record_tc_dset)
Record processing entry for raw bo.
- **epicsExportAddress** (dset, longoutval_record_tc_dset)
Record processing entry for longout.
- **epicsExportAddress** (dset, mbboaval_record_tc_dset)
Record processing entry for mbbo.
- **epicsExportAddress** (dset, mbborval_record_tc_dset)
Record processing entry for raw mbbo.
- **epicsExportAddress** (dset, mbboDirectval_record_tc_dset)
Record processing entry for mbboDirect.
- **epicsExportAddress** (dset, mbboDirectrval_record_tc_dset)
Record processing entry for raw mbboDirect.
- **epicsExportAddress** (dset, stringoutval_record_tc_dset)
Record processing entry for stringout.
- **epicsExportAddress** (dset, lsoval_record_tc_dset)
Record processing entry for lso (long string)

Variables

- callbackQueueStats **DevTc::callback_status {}**

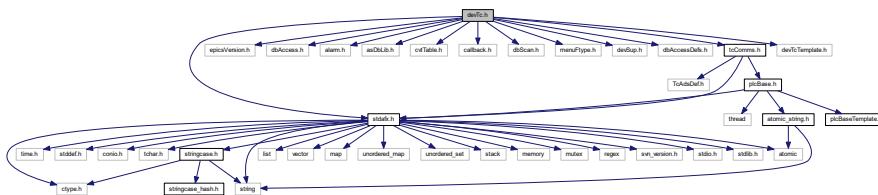
9.2.1 Detailed Description

Methods for TwinCAT/ADS device support.

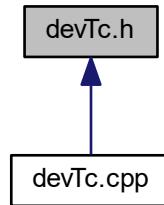
9.3 devTc.h File Reference

```
#include "stdafx.h"
#include "epicsVersion.h"
#include "dbAccess.h"
#include "alarm.h"
#include "asDbLib.h"
#include "cvtTable.h"
#include "callback.h"
#include "dbScan.h"
#include "menuFtype.h"
```

```
#include "devSup.h"
#include "dbAccessDefs.h"
#include "tcComms.h"
#include "devTcTemplate.h"
Include dependency graph for devTc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DevTc::register_devsup](#)
Device support registration.
- class [DevTc::EpicsInterface](#)
Epics interface class.
- struct [DevTc::epics_record_traits< RecType >](#)
Epics record traits.
- struct [DevTc::devTcDeflo< RecType >](#)
Device support record.
- struct [DevTc::devTcDefIn< RecType >](#)
Device support input record.
- struct [DevTc::devTcDefOut< RecType >](#)
device support output record.
- struct [DevTc::devTcDefWaveformIn< RecType >](#)
device support waveform record.

Namespaces

- [DevTc](#)
Namespace for TCat device support.

Enumerations

- enum DevTc::epics_record_enum {
 DevTc::epics_record_enum::aaival = 0, DevTc::epics_record_enum::aaoval, DevTc::epics_record_enum::aival,
 DevTc::epics_record_enum::aoval,
 DevTc::epics_record_enum::bival, DevTc::epics_record_enum::boval, DevTc::epics_record_enum::eventval,
 DevTc::epics_record_enum::histogramval,
 DevTc::epics_record_enum::longinval, DevTc::epics_record_enum::longoutval, DevTc::epics_record_enum::mbbival,
 DevTc::epics_record_enum::mbboval,
 DevTc::epics_record_enum::mbbiDirectval, DevTc::epics_record_enum::mbboDirectval, DevTc::epics_record_enum::stringinval,
 DevTc::epics_record_enum::stringoutval,
 DevTc::epics_record_enum::lsival, DevTc::epics_record_enum::lsoval, DevTc::epics_record_enum::waveformval,
 DevTc::epics_record_enum::airval,
 DevTc::epics_record_enum::aorval, DevTc::epics_record_enum::birval, DevTc::epics_record_enum::borval,
 DevTc::epics_record_enum::mbbiDirectval,
 DevTc::epics_record_enum::mbboDirectval, DevTc::epics_record_enum::mbbirval, DevTc::epics_record_enum::mbborval,
 DevTc::epics_record_enum::epics_record_enumEnd,
 DevTc::epics_record_enum::invalidval = -1 }

Epics record type enum.

Functions

- const std::regex DevTc::tc_regex ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +:(8[0-9][0-9]))/(\\d{1,9})/(\\d{1,9}):\\d{1,9}"))

Callback for output record.
- const std::regex DevTc::info_regex ("((tc):://((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?) +:(8[0-9][0-9]))/(info)/([A-Za-z0-9_]+)"))

Regex for indentifying info records.
- void DevTc::complete_io_scan (EpicsInterface *epics, IOSCANPVT ioscan, int prio) noexcept

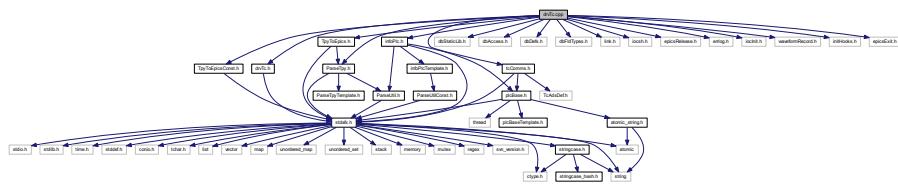
9.3.1 Detailed Description

Header which includes classes for TwinCAT/ADS device support.

9.4 drvTc.cpp File Reference

```
#include "drvTc.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "TpyToEpicsConst.h"
#include "infoPlc.h"
#include "dbStaticLib.h"
#include "dbAccess.h"
#include "dbDefs.h"
#include "dbFldTypes.h"
#include "link.h"
#include "iocsh.h"
#include "epicsRelease.h"
#include "errlog.h"
#include "iocInit.h"
#include "waveformRecord.h"
#include "initHooks.h"
```

```
#include "tcComms.h"
#include "epicsExit.h"
Include dependency graph for drvTc.cpp:
```



Classes

- class DevTc::epics_tc_db_processing
EPICS/TCat db processing.

Namespaces

- **DevTc**
Namespace for TCat device support.

Typedefs

- using `DevTc::filename_rule_list_tuple` = `std::tuple< std::stringcase, std::stringcase, epics_list_processing *, bool >`
Tuple for filnemae, rule and list processing.
 - using `DevTc::tc_listing_def` = `std::vector< filename_rule_list_tuple >`
List of tuples for filnemae, rule and list processing.
 - using `DevTc::dirname_arg_macro_tuple` = `std::tuple< std::stringcase, std::stringcase, epics_macrofiles_processing *, const char * >`
Tuple for directory name, argument and macro list processing.
 - using `DevTc::tc_macro_def` = `std::vector< dirname_arg_macro_tuple >`
List of tuples for directory name, argument and macro list processing.

Functions

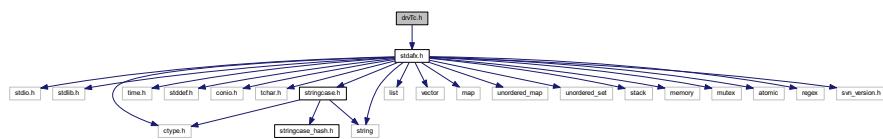
- void **DevTc::tcLoadRecords** (const iocshArgBuf *args)
Load TwinCAT records.
 - void **DevTc::tcSetScanRate** (const iocshArgBuf *args) noexcept
Set the scan rate.
 - void **DevTc::tcList** (const iocshArgBuf *args)
Generate channel lists.
 - void **DevTc::tcMacro** (const iocshArgBuf *args)
Generated macro files.
 - void **DevTc::tcAlias** (const iocshArgBuf *args)
Define alias and replacement rules.
 - void **DevTc::tcInfoPrefix** (const iocshArgBuf *args)
Sets the info prefix.
 - void **DevTc::tcPrintVals** (const iocshArgBuf *args)
Print all values.
 - void **DevTc::tcPrintVal** (const iocshArgBuf *args)
Print value.

9.4.1 Detailed Description

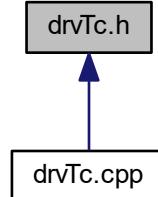
This contains functions for driver support for EPICS. These routines are used during the IOC initialization, and allow the IOC to initialize the TwinCAT interface during EPICS initialization.

9.5 drvTc.h File Reference

```
#include "stdafx.h"
Include dependency graph for drvTc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DevTc::tcRegisterToLocShell](#)
Register TC commands.

Namespaces

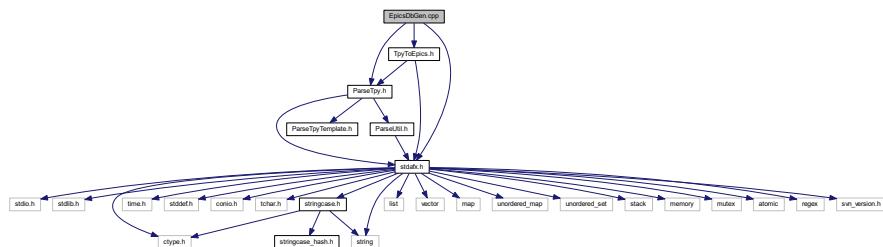
- [DevTc](#)
Namespace for TCat device support.

9.5.1 Detailed Description

Header which includes classes for accessing TwinCAT/ADS from EPICS.

9.6 EpicsDbGen.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
Include dependency graph for EpicsDbGen.cpp:
```



Functions

- int main (int argc, char *argv[])

9.6.1 Detailed Description

Source for the main program that generates an EPICs .db file

9.6.2 Function Documentation

9.6.2.1 main()

```
int main ( int argc,  
           char * argv[ ] )
```

Main program

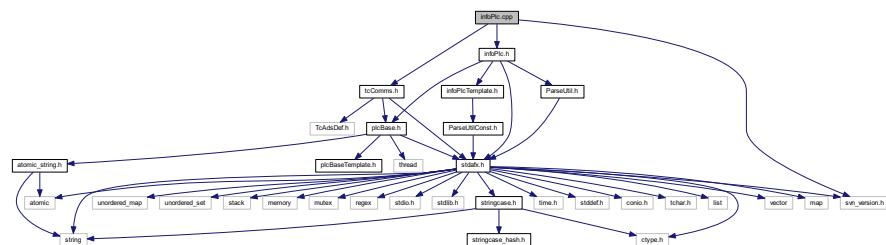
Definition at line 17 of file EpicsDbGen.cpp.

References EpicsTpy::split_io_support::flush(), EpicsTpy::epics_macrofiles_processing::flush(), EpicsTpy::split_io_support::get_file(), EpicsTpy::multi_io_support::get_filein_total(), EpicsTpy::multi_io_support::get_fileout_total(), EpicsTpy::multi_io_support::get_indirname(), EpicsTpy::multi_io_support::get_outdirname(), EpicsTpy::split_io_support::get_processed_io(), EpicsTpy::split_io_support::get_processed_READONLY(), EpicsTpy::split_io_support::get_processed_total(), EpicsTpy::epics_macrofiles_processing::get_processed_total(), ParseTpy::tpy_file::get_project_info(), ParseUtil::replacement_rules::get_rule_table(), ParseTpy::compiler_info::get_tcat_version_major(), ParseUtil::tag_processing::getopt(), ParseTpy::tpy_file::parse(), ParseTpy::tpy_file::process_symbols(), EpicsTpy::multi_io_support::set_indirname(), ParseUtil::replacement_rules::set_rule_table(), and EpicsTpy::epics_macrofiles_processing::set_twincat3().

Here is the call graph for this function:

9.7 infoPlc.cpp File Reference

```
#include "infoPlc.h"
#include "tcComms.h"
#include "svn_version.h"
Include dependency graph for infoPlc.cpp:
```



Namespaces

- **InfoPlc**
Namespace for Info communication.

Variables

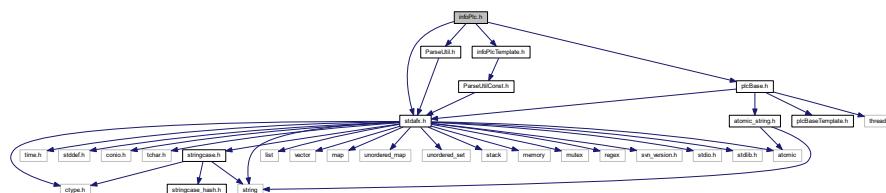
- constexpr time_t **InfoPlc::WINDOWS_TICK** = 10'000'000
- constexpr time_t **InfoPlc::SEC_TO_UNIX_EPOCH** = 11'644'473'600LL

9.7.1 Detailed Description

Defines methods for the info PLC.

9.8 infoPlc.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "plcBase.h"
#include "infoPlcTemplate.h"
Include dependency graph for infoPlc.h:
```



This graph shows which files directly or indirectly include this file:

Classes

- class [InfoPlc::InfoInterface](#)
Info interface.
- class [InfoPlc::process_arg_info](#)
Arguments for processing.

Namespaces

- [InfoPlc](#)
Namespace for Info communication.

Typedefs

- using [InfoPlc::info_update_method](#) = bool(InfoInterface::*)()
Pointer to info update method.
- using [InfoPlc::info_dbrecord_type](#) = std::tuple< ParseUtil::variable_name, ParseUtil::process_type_enum, ParseUtil::opc_list, std::stringcase, bool, update_enum, info_update_method >
- using [InfoPlc::info_dbrecord_list](#) = std::vector<info_dbrecord_type>
List type of db info tuples.

Enumerations

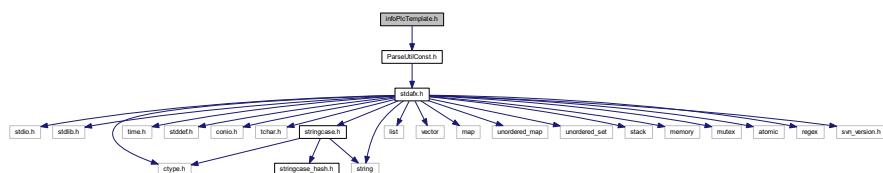
- enum [InfoPlc::update_enum](#) { [InfoPlc::update_enum::forever](#), [InfoPlc::update_enum::once](#), [InfoPlc::update_enum::done](#) }
Update frequency type.

9.8.1 Detailed Description

Header which includes classes for the info PLC.

9.9 infoPlcTemplate.h File Reference

```
#include "ParseUtilConst.h"
Include dependency graph for infoPlcTemplate.h:
```



This graph shows which files directly or indirectly include this file:

Namespaces

- [InfoPlc](#)

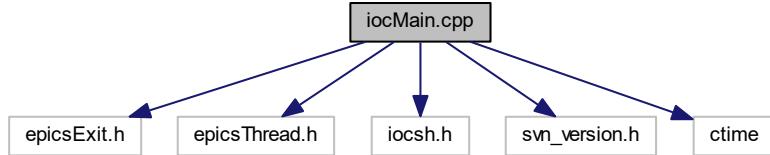
Namespace for Info communication.

9.9.1 Detailed Description

Header which includes classes for the info PLC.

9.10 iocMain.cpp File Reference

```
#include "epicsExit.h"
#include "epicsThread.h"
#include "iocsh.h"
#include "svn_version.h"
#include <ctime>
Include dependency graph for iocMain.cpp:
```



Functions

- [__declspec \(dllimport\) void stopTc\(void\)](#)
DLL import for stopTc.
- int [main \(int argc, char *argv\[\]\)](#)
tcloc

9.10.1 Detailed Description

The main program for the TwinCAT IOC.

9.10.2 Function Documentation

9.10.2.1 main()

```
int main (
    int argc,
    char * argv[] )
```

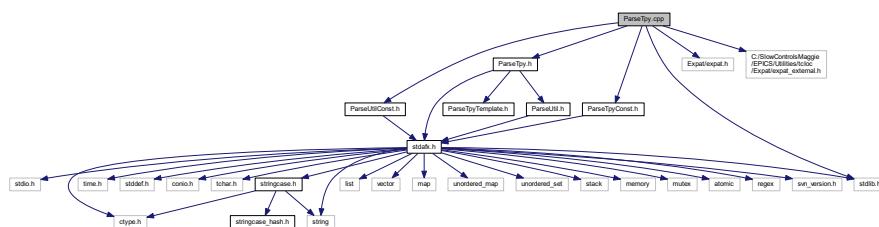
tcloc

Main program for tcloc

Definition at line 25 of file iocMain.cpp.

9.11 ParseTpy.cpp File Reference

```
#include "ParseTpy.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "Expat/expat.h"
Include dependency graph for ParseTpy.cpp:
```



Classes

- class [ParseTpy::parserinfo_type](#)

Parser information.

Namespaces

- [ParseTpy](#)

Namespace for parsing.

Macros

- `#define XML_STATIC`
Static linking.
- `#define XML_FMT_INT_MOD "I"`
Int format.

Functions

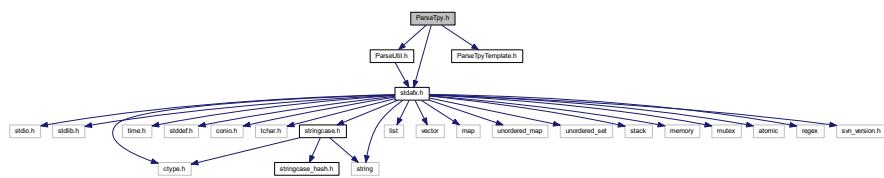
- bool `ParseTpy::compareNamesWoNamespace` (const `std::stringcase` &p1, const `std::stringcase` &p2)
- bool `ParseTpy::get_decoration` (const char **atts, unsigned int &decoration)
- bool `ParseTpy::get_pointer` (const char **atts)

9.11.1 Detailed Description

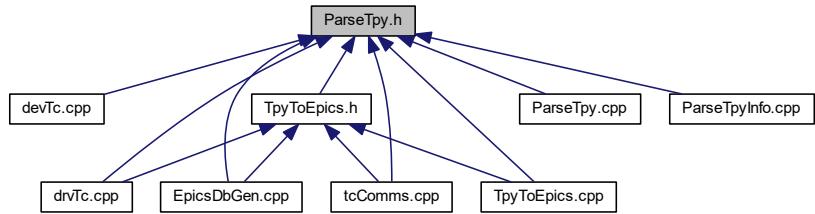
Methods to parse TwinCAT tpy file.

9.12 ParseTpy.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "ParseTpyTemplate.h"
Include dependency graph for ParseTpy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `ParseTpy::ads_routing_info`
ADS routing information.
- class `ParseTpy::compiler_info`
Compiler information.
- class `ParseTpy::project_record`
Project information.
- class `ParseTpy::base_record`
Base record definition.

- class [ParseTpy::item_record](#)
item record
- class [ParseTpy::type_record](#)
Type record information.
- class [ParseTpy::type_map](#)
Type dictionary.
- class [ParseTpy::symbol_record](#)
Symbol record.
- class [ParseTpy::tpy_file](#)
Tpy file parsing.

Namespaces

- [ParseTpy](#)
Namespace for parsing.

Typedefs

- using [ParseTpy::dimension](#) = std::pair< int, int >
- using [ParseTpy::dimensions](#) = std::list< dimension >
- using [ParseTpy::enum_map](#) = std::map< int, std::stringcase >
- using [ParseTpy::enum_pair](#) = std::pair< int, std::stringcase >
- using [ParseTpy::item_list](#) = std::list< item_record >
- using [ParseTpy::type_multimap](#) = std::multimap< unsigned int, type_record >
- using [ParseTpy::symbol_list](#) = std::list< symbol_record >

Enumerations

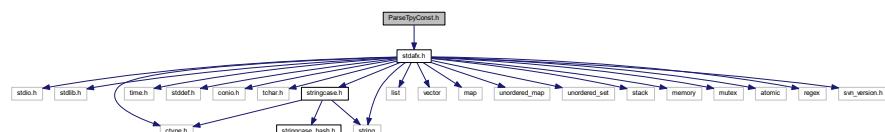
- enum [ParseTpy::type_enum](#) {
 [ParseTpy::type_enum::unknown](#), [ParseTpy::type_enum::simple](#), [ParseTpy::type_enum::arraytype](#), [ParseTpy::type_enum::enumtype](#),
[ParseTpy::type_enum::structtype](#), [ParseTpy::type_enum::functionblock](#) }
- Type enum.*

9.12.1 Detailed Description

Header which includes classes to parse a TwinCAT tpy file. It includes "ParseUtil.h" and "ParseTpyTemplate.h"

9.13 ParseTpyConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseTpyConst.h:
```



This graph shows which files directly or indirectly include this file:

Namespaces

- [ParseUtil](#)
Namespace for parsing utilities.
- [ParseTpy](#)
Namespace for parsing.

Variables

- `const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"`
PLC project info.
- `const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"`
Project info.
- `const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"`
Routing info.
- `const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"`
Compiler info.
- `const char *const ParseTpy::xmlAdsInfo = "AdsInfo"`
ADS info.
- `const char *const ParseTpy::xmlDataTypes = "DataTypes"`
Data types.
- `const char *const ParseTpy::xmlDataType = "DataType"`
Data type.
- `const char *const ParseTpy::xmlSymbols = "Symbols"`
Symbols.
- `const char *const ParseTpy::xmlSymbol = "Symbol"`
Symbol.
- `const char *const ParseTpy::xmlProperties = "Properties"`
Properties.
- `const char *const ParseTpy::xmlProperty = "Property"`
Property.
- `const char *const ParseTpy::xmlCompilerVersion = "CompilerVersion"`
Compiler version.
- `const char *const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"`
TwinCAT version.
- `const char *const ParseTpy::xmlCpuFamily = "CpuFamily"`
CPU family.
- `const char *const ParseTpy::xmlNetId = "NetId"`
Net ID.
- `const char *const ParseTpy::xmlPort = "Port"`
Port.
- `const char *const ParseTpy::xmlTargetName = "TargetName"`
Target name.
- `const char *const ParseTpy::xmlName = "Name"`
Name.
- `const char *const ParseTpy::xmlType = "Type"`
Type.
- `const char *const ParseTpy::xmlAttrDecoration = "Decoration"`
Decoration.
- `const char *const ParseTpy::xmlAttrPointer = "Pointer"`

- const char *const ParseTpy::xmlIGroup = "IGroup"

I Group.
- const char *const ParseTpy::xmlIOffset = "IOffset"

I Offset.
- const char *const ParseTpy::xmlBitSize = "BitSize"

Bit size.
- const char *const ParseTpy::xmlBitOffs = "BitOffs"

Bit Offset.
- const char *const ParseTpy::xmlArrayInfo = "ArrayInfo"

Array info.
- const char *const ParseTpy::xmlArrayLBound = "LBound"

Lower bound.
- const char *const ParseTpy::xmlArrayElements = "Elements"

Elements.
- const char *const ParseTpy::xmlSubItem = "SubItem"

Sub item.
- const char *const ParseTpy::xmlFbInfo = "FbInfo"

Fb info.
- const char *const ParseTpy::xmlEnumInfo = "EnumInfo"

Enum info.
- const char *const ParseTpy::xmlEnumText = "Text"

Text.
- const char *const ParseTpy::xmlEnumEnum = "Enum"

Enum.
- const char *const ParseTpy::xmlEnumComment = "Comment"

Comment.
- const char *const ParseTpy::xmlValue = "Value"

Value.
- const char *const ParseTpy::xmlDesc = "Desc"

Description.
- const char *const ParseTpy::opcExport = "opc"

OPC.
- const char *const ParseTpy::opcProp = "opc_prop"

OPC property.
- const char *const ParseTpy::opcBracket = "["

OPC bracket.
- const char *const ParseTpy::xmlSubstitution = "TcSubstitution"

Substitution.
- const char *const ParseTpy::xmlAlias = "Alias"

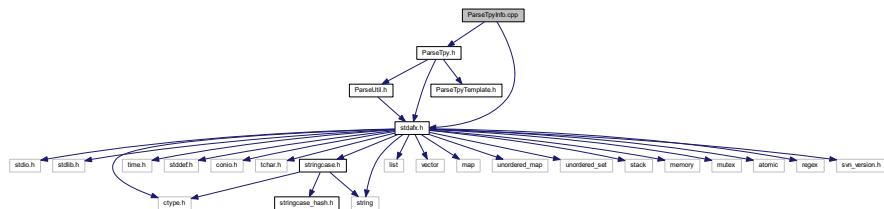
Substitution EPICS channel name.

9.13.1 Detailed Description

Header which includes constants needed to parse a TwinCAT tpy file.

9.14 ParseTpyInfo.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
Include dependency graph for ParseTpyInfo.cpp:
```



Classes

- class [syminfo_processing](#)

Symbol processing.

Functions

- int [main](#) (int argc, const char *argv[])
- tpyinfo*

9.14.1 Detailed Description

Source file for tpy parsing methods.

9.14.2 Function Documentation

9.14.2.1 main()

```
int main (
    int argc,
    const char * argv[ ] )
```

tpyinfo

Main program for tpyinfo

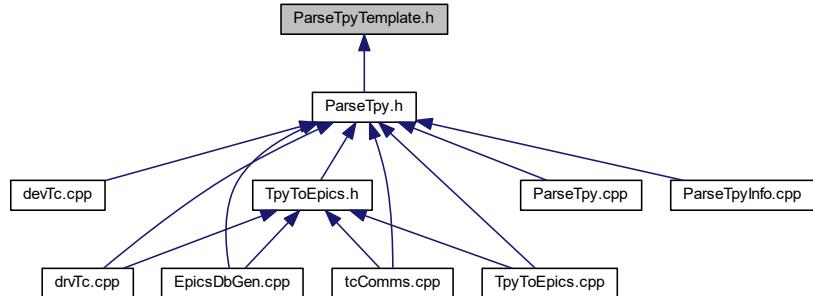
Definition at line 93 of file ParseTpyInfo.cpp.

References `ParseUtil::tag_processing::getopt()`, `ParseTpy::tpy_file::parse()`, and `ParseTpy::tpy_file::process_< symbols()`.

Here is the call graph for this function:

9.15 ParseTpyTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpy](#)
Namespace for parsing.

9.15.1 Detailed Description

Templates for functions for parsing tpy file.

9.16 ParseUtil.cpp File Reference

```
#include "ParseUtil.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "Expat/expat.h"
```

Include dependency graph for ParseUtil.cpp:

Classes

- class [ParseUtil::parserinfo_type](#)
Parser information.

Namespaces

- [ParseUtil](#)
Namespace for parsing utilities.

Macros

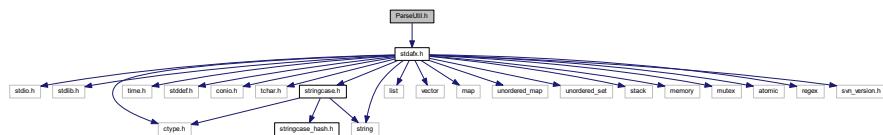
- `#define XML_STATIC`
Static linking.
- `#define XML_FMT_INT_MOD "I"`
Int format.

9.16.1 Detailed Description

Utility methods to parsing.

9.17 ParseUtil.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseUtil.h:
```



This graph shows which files directly or indirectly include this file:

Classes

- class `ParseUtil::replacement_rules`
Replacement rules.
- class `ParseUtil::optarg`
Optional arguments.
- class `ParseUtil::opc_list`
OPC list.
- class `ParseUtil::variable_name`
Variable name.
- class `ParseUtil::bit_location`
Bit location.
- class `ParseUtil::memory_location`
Memory location.
- class `ParseUtil::process_arg`
Arguments for processing.
- class `ParseUtil::process_arg_tc`
Arguments for processing.
- class `ParseUtil::tag_processing`
Tag processing selection.
- class `ParseUtil::substitution`
Channel substitution.
- class `ParseUtil::substitution_list`
List of channel substitutions.

Namespaces

- `ParseUtil`

Namespace for parsing utilities.

Typedefs

- using `ParseUtil::replacement_table` = `std::map< std::stringcase, std::stringcase >`
- using `ParseUtil::property_map` = `std::map< int, std::stringcase >`
- using `ParseUtil::property_el` = `std::pair< int, std::stringcase >`
- using `ParseUtil::substitution_map` = `std::map< std::stringcase, substitution >`

Map of channel substitution.

Enumerations

- enum `ParseUtil::opc_enum` { `ParseUtil::opc_enum::no_change`, `ParseUtil::opc_enum::publish`, `ParseUtil::opc_enum::silent` }
- *OPC state enum.*
- enum `ParseUtil::process_type_enum` {
`ParseUtil::process_type_enum::pt_invalid`, `ParseUtil::process_type_enum::pt_int`, `ParseUtil::process_type_enum::pt_real`,
`ParseUtil::process_type_enum::pt_bool`,
`ParseUtil::process_type_enum::pt_string`, `ParseUtil::process_type_enum::pt_enum`, `ParseUtil::process_type_enum::pt_binary` }
- *Process type.*
- enum `ParseUtil::process_tag_enum` { `ParseUtil::process_tag_enum::all`, `ParseUtil::process_tag_enum::atomic`,
`ParseUtil::process_tag_enum::structured` }
- *Tag processing enum.*
- enum `ParseUtil::process_substitution_enum` { `ParseUtil::process_substitution_enum::standard`, `ParseUtil::process_substitution_enum::ignore` }
- *Substitution processing enum.*

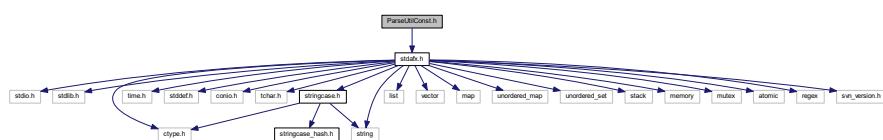
9.17.1 Detailed Description

Header which includes utility classes for parsing. It includes "ParseUtilTemplate.h"

9.18 ParseUtilConst.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for ParseUtilConst.h:



This graph shows which files directly or indirectly include this file:

Namespaces

- [ParseTyp](#)
Namespace for parsing.
- [ParseUtil](#)
Namespace for parsing utilities.

Variables

- `constexpr int ParseUtil::OPC_PROP_CDT = 1`
- `constexpr int ParseUtil::OPC_PROP_VALUE = 2`
- `constexpr int ParseUtil::OPC_PROP_QUALITY = 3`
- `constexpr int ParseUtil::OPC_PROP_TIME = 4`
- `constexpr int ParseUtil::OPC_PROP_RIGHTS = 5`
- `constexpr int ParseUtil::OPC_PROP_SCANRATE = 6`
- `constexpr int ParseUtil::OPC_PROP_UNIT = 100`
- `constexpr int ParseUtil::OPC_PROP_DESC = 101`
- `constexpr int ParseUtil::OPC_PROP_HIEU = 102`
- `constexpr int ParseUtil::OPC_PROP_LOEU = 103`
- `constexpr int ParseUtil::OPC_PROP_HIRANGE = 104`
- `constexpr int ParseUtil::OPC_PROP_LORANGE = 105`
- `constexpr int ParseUtil::OPC_PROP_CLOSE = 106`
- `constexpr int ParseUtil::OPC_PROP_OPEN = 107`
- `constexpr int ParseUtil::OPC_PROP_TIMEZONE = 108`
- `constexpr int ParseUtil::OPC_PROP_FGC = 201`
- `constexpr int ParseUtil::OPC_PROP_BGC = 202`
- `constexpr int ParseUtil::OPC_PROP_BLINK = 203`
- `constexpr int ParseUtil::OPC_PROP_BMP = 204`
- `constexpr int ParseUtil::OPC_PROP SND = 205`
- `constexpr int ParseUtil::OPC_PROP_HTML = 206`
- `constexpr int ParseUtil::OPC_PROP_AVI = 207`
- `constexpr int ParseUtil::OPC_PROP_ALMSTAT = 300`
- `constexpr int ParseUtil::OPC_PROP_ALMHELP = 301`
- `constexpr int ParseUtil::OPC_PROP_ALMAREAS = 302`
- `constexpr int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303`
- `constexpr int ParseUtil::OPC_PROP_ALMCONDITION = 304`
- `constexpr int ParseUtil::OPC_PROP_ALMLIMIT = 305`
- `constexpr int ParseUtil::OPC_PROP_ALMDB = 306`
- `constexpr int ParseUtil::OPC_PROP_ALMH = 307`
- `constexpr int ParseUtil::OPC_PROP_ALMH = 308`
- `constexpr int ParseUtil::OPC_PROP_ALML = 309`
- `constexpr int ParseUtil::OPC_PROP_ALMLL = 310`
- `constexpr int ParseUtil::OPC_PROP_ALMROC = 311`
- `constexpr int ParseUtil::OPC_PROP_ALMDEV = 312`
- `constexpr int ParseUtil::OPC_PROP_PREC = 8500`
- `constexpr int ParseUtil::OPC_PROP_ZRST = 8510`
- `constexpr int ParseUtil::OPC_PROP_FFST = 8525`
- `constexpr int ParseUtil::OPC_PROP_RECTYPE = 8600`
- `constexpr int ParseUtil::OPC_PROP_INOUT = 8601`
- `const char *const ParseUtil::OPC_PROP_INPUT = "input"`
- `const char *const ParseUtil::OPC_PROP_OUTPUT = "output"`
- `constexpr int ParseUtil::OPC_PROP_TSE = 8602`
- `constexpr int ParseUtil::OPC_PROP_PINI = 8603`

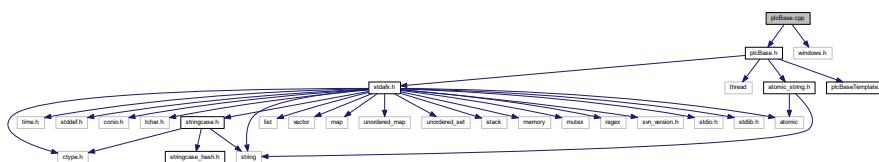
- `constexpr int ParseUtil::OPC_PROP_DTYP = 8604`
- `constexpr int ParseUtil::OPC_PROP_SERVER = 8610`
- `constexpr int ParseUtil::OPC_PROP_PLNAME = 8611`
- `constexpr int ParseUtil::OPC_PROP_ALIAS = 8620`
- `const char *const ParseUtil::OPC_NAME_ALIAS = "ALIAS"`
- `constexpr int ParseUtil::OPC_PROP_ALMOSV = 8700`
- `constexpr int ParseUtil::OPC_PROP_ALMZSV = 8701`
- `constexpr int ParseUtil::OPC_PROP_ALMCOSV = 8702`
- `constexpr int ParseUtil::OPC_PROP_ALMUNSV = 8703`
- `constexpr int ParseUtil::OPC_PROP_ALMZRSV = 8710`
- `constexpr int ParseUtil::OPC_PROP_ALMFFSV = 8725`
- `constexpr int ParseUtil::OPC_PROP_ALMHHSV = 8727`
- `constexpr int ParseUtil::OPC_PROP_ALMHHSV = 8728`
- `constexpr int ParseUtil::OPC_PROP_ALMLSV = 8729`
- `constexpr int ParseUtil::OPC_PROP_ALMLLSV = 8730`
- `constexpr int ParseUtil::OPC_PROP_FIELD_BEG = 8800`
- `constexpr int ParseUtil::OPC_PROP_FIELD_END = 9000`

9.18.1 Detailed Description

Header which includes OPC constants.

9.19 plcBase.cpp File Reference

```
#include "plcBase.h"
#include <windows.h>
Include dependency graph for plcBase.cpp:
```



Namespaces

- `plc`
Namespace for abstract plc functionality.

Typedefs

- `using plc::scanner_thread_args = struct { plc::BasePLC *plc
Scanner thread arguments.`

Functions

- template<> bool [plc::reset_and_read](#) (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, const DataValueTypeDef::atomic_string *source) noexcept
 - Reset and read.*
 - template<> bool [plc::write_and_test](#) (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source) noexcept
 - Write and test.*
 - VOID CALLBACK [plc::ScannerProc](#) (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)
 - Scanner thread callback.*
 - DWORD WINAPI [plc::scannerThread](#) (scanner_thread_args args) noexcept
 - Scanner thread.*
 - [__declspec](#) (dllexport) void stopTc(void)
 - Stop TwinCAT.*

Variables

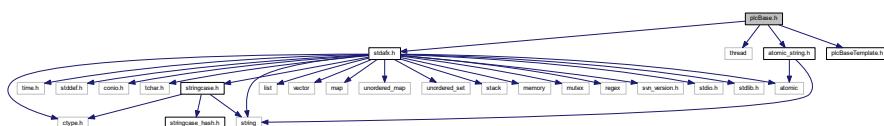
- long `plc::scannerperiod`
Period in ms of the scanner.
 - `plc::BasePLC::scanner_func plc::scanner`
Address of the scanner function to be used (read, write, or update)

9.19.1 Detailed Description

Defines methods for the internal record entry.

9.20 plcBase.h File Reference

```
#include "stdafx.h"
#include <thread>
#include "atomic_string.h"
#include "plcBaseTemplate.h"
Include dependency graph for plcBase.h:
```



This graph shows which files directly or indirectly include this file:

Classes

- class `plc::Interface`
Abstract interface.
- struct `plc::DataValueTraits< T >`
Data value traits.
- struct `plc::DataValueTypeDef`
Collection of type definitions.
- class `plc::DataValue`
Data value.
- class `plc::BaseRecord`
Class for managing a tag/channel.
- class `plc::BasePLC`
Base PLC.
- class `plc::System`
System to keep track of PLCs.

Namespaces

- `plc`
Namespace for abstract plc functionality.

Typedefs

- using `plc::BasePLCPtr = std::shared_ptr< BasePLC >`
Smart pointer to PLC.
- using `plc::InterfacePtr = std::unique_ptr< Interface >`
Smart pointer to interface.
- using `plc::BaseRecordPtr = std::shared_ptr< BaseRecord >`
smart pointer to record
- using `plc::BaseRecordList = std::unordered_map< std::stringcase, BaseRecordPtr >`
list of record
- using `plc::BasePLCList = std::map< std::stringcase, BasePLCPtr >`
BasePLC map.

Enumerations

- enum `plc::data_type_enum {`
`plc::data_type_enum::dtInvalid, plc::data_type_enum::dtBool, plc::data_type_enum::dtInt8, plc::data_type_enum::dtUInt8,`
`plc::data_type_enum::dtInt16, plc::data_type_enum::dtUInt16, plc::data_type_enum::dtInt32, plc::data_type_enum::dtUInt32,`
`plc::data_type_enum::dtInt64, plc::data_type_enum::dtUInt64, plc::data_type_enum::dtFloat, plc::data_type_enum::dtDouble,`
`plc::data_type_enum::dtString, plc::data_type_enum::dtWString, plc::data_type_enum::dtBinary }`
Data type enumeration.
- enum `plc::access_rights_enum { plc::access_rights_enum::read_only, plc::access_rights_enum::write_only,`
`plc::access_rights_enum::read_write }`
Access rights enum.

9.20.1 Detailed Description

Header which includes abstract base classes for defining an internal record entry for the IOC.

9.21 plcBaseTemplate.h File Reference

This graph shows which files directly or indirectly include this file:

Namespaces

- **plc**

Namespace for abstract plc functionality.

Functions

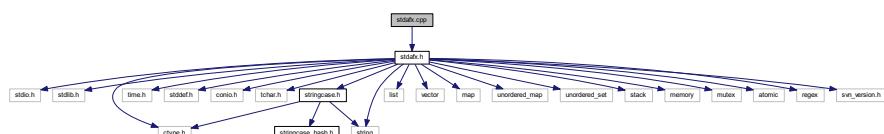
- template<typename T , typename U >
bool **plc::reset_and_read** (DataValueTypeDef::atomic_bool &dirty, T &dest, U source) noexcept
Reset and read.
- template<typename T >
bool **plc::reset_and_read** (DataValueTypeDef::atomic_bool &dirty, T &dest, const typename DataValueTraits< T >::traits_atomic *source) noexcept
Reset and read.
- template<typename T , typename U >
bool **plc::write_and_test** (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, U dest, const T &source) noexcept
Write and test.
- template<typename T >
bool **plc::write_and_test** (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest, const T &source) noexcept
Write and test.

9.21.1 Detailed Description

Header which includes templated methods for abstract record class and the DataValue class.

9.22 stdafx.cpp File Reference

```
#include "stdafx.h"
Include dependency graph for stdafx.cpp:
```



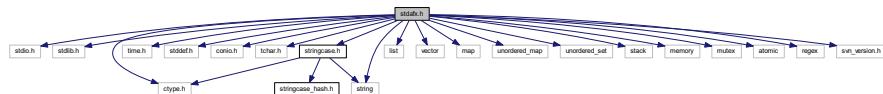
9.22.1 Detailed Description

source file that includes just the standard includes EpicsDbGen.pch will be the pre-compiled header stdafx.obj will contain the pre-compiled type information

9.23 stdafx.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <stddef.h>
#include <conio.h>
#include <tchar.h>
#include <string>
#include "stringcase.h"
#include <list>
#include <vector>
#include <map>
#include <unordered_map>
#include <unordered_set>
#include <stack>
#include <memory>
#include <mutex>
#include <atomic>
#include <regex>
#include "svn_version.h"
```

Include dependency graph for stdafx.h:



This graph shows which files directly or indirectly include this file:

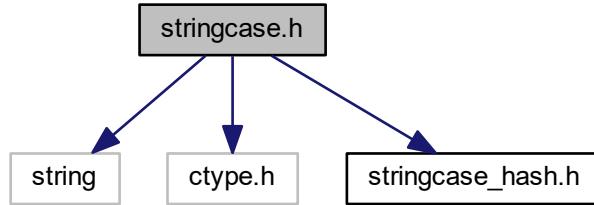
9.23.1 Detailed Description

Include file for standard system include files, or project specific include files that are used frequently, but are changed infrequently

9.24 stringcase.h File Reference

```
#include <string>
#include <ctype.h>
```

```
#include "stringcase_hash.h"
Include dependency graph for stringcase.h:
```



This graph shows which files directly or indirectly include this file:

Classes

- struct `std::case_char_traits`
case insensitive traits.
- struct `std::case_wchar_traits`
case insensitive unicode traits.

Typedefs

- using `std::stringcase = std::basic_string< char, case_char_traits >`
case insensitive string.
- using `std::wstringcase = std::basic_string< wchar_t, case_wchar_traits >`
case insensitive string.

Functions

- int `std::strncasecmp (const char *s1, const char *s2, size_t n) noexcept`
case insensitive compare with maximum length
- int `std::wcscasecmp (const wchar_t *s1, const wchar_t *s2, size_t n) noexcept`
case insensitive unicode compare with maximum length
- void `std::trim_space (std::stringcase &s)`
- void `std::trim_space (std::wstringcase &s)`
- template<class Container , class String , class Predicate >
`void std::split_string (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)`
Splits a strings.

9.24.1 Detailed Description

Functions and classes for a case-insensitive string.

9.25 stringcase_hash.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct `std::std::hash< std::stringcase >`
hash for case insensitive string.
- struct `std::std::hash< std::wstringcase >`
hash for case insensitive unicode string.

9.25.1 Detailed Description

Specialization for case sensitive strings.

9.26 tcComms.cpp File Reference

```
#include "tcComms.h"
#include "infoPlc.h"
#include "ParseTpy.h"
#include "windows.h"
#include "TpyToEpics.h"
#include "TcAdsDef.h"
#include "TcAdsAPI.h"
#include <memory>
#include <filesystem>
#include <chrono>
Include dependency graph for tcComms.cpp:
```

Namespaces

- `TcComms`
Namespace for TCat communication.

Functions

- void `TcComms::errorPrintf` (int nErr) noexcept
`errorPrintf`
- void __stdcall `TcComms::ADSCallback` (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plcd)
- void __stdcall `TcComms::RouterCall` (long nReason)

Variables

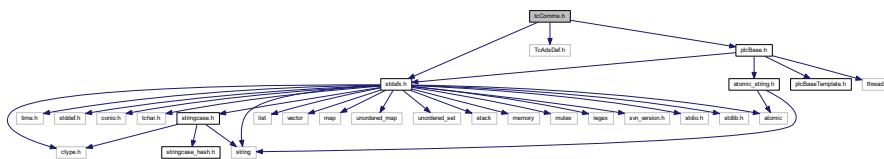
- constexpr time_t `TcComms::WINDOWS_TICK` = 10'000'000
- constexpr time_t `TcComms::SEC_TO_UNIX_EPOCH` = 11'644'473'600LL

9.26.1 Detailed Description

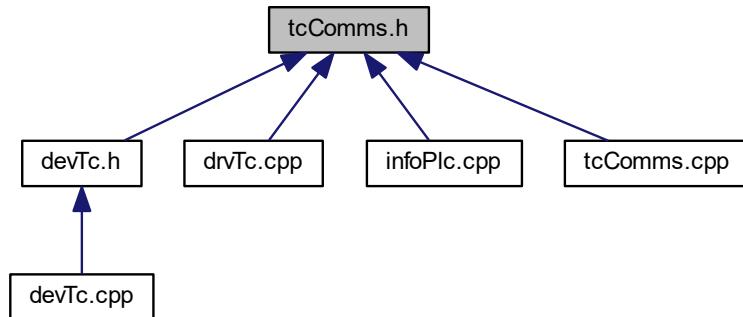
Defines methods for TwinCAT communication.

9.27 tcComms.h File Reference

```
#include "stdafx.h"
#include <TcAdsDef.h>
#include "plcBase.h"
Include dependency graph for tcComms.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct [TcComms::DataPar](#)
Memory location struct.
- class [TcComms::TCatInterface](#)
TCat interface class.
- class [TcComms::tcProcWrite](#)
TwinCAT process write requests.
- class [TcComms::TcPLC](#)
TwinCAT PLC.
- class [TcComms::AmsRouterNotification](#)
AMS Router Notification.

Namespaces

- **TcComms**

Namespace for TCat communication.

Variables

- `constexpr int TcComms::MAX_REQ_SIZE = 250000`
maximum allowed request size (bytes)
- `constexpr int TcComms::MAX_SINGLE_GAP_SIZE = 50`
maximum allowed size (bytes) of a memory gap within continuous request
- `constexpr double TcComms::MAX_REL_GAP = 0.25`
(maximum allowed total gap size) / (current request size)
- `constexpr int TcComms::MIN_REL_GAP_SIZE = 100`
minimum allowed relative gap size (bytes)
- `constexpr int TcComms::default_scanrate = 100`
default PLC TwinCAT scan rate (100ms)
- `constexpr int TcComms::minimum_scanrate = 5`
minimum PLC TwinCAT scan rate (5ms)
- `constexpr int TcComms::maximum_scanrate = 10000`
maximum PLC TwinCAT scan rate (10s)
- `constexpr int TcComms::default_multiple = 10`
default multiple for PLC EPICS scan rate (10)
- `constexpr int TcComms::minimum_multiple = 1`
minimum multiple for PLC EPICS scan rate (1)
- `constexpr int TcComms::maximum_multiple = 200`
maximum multiple for PLC EPICS scan rate (200)

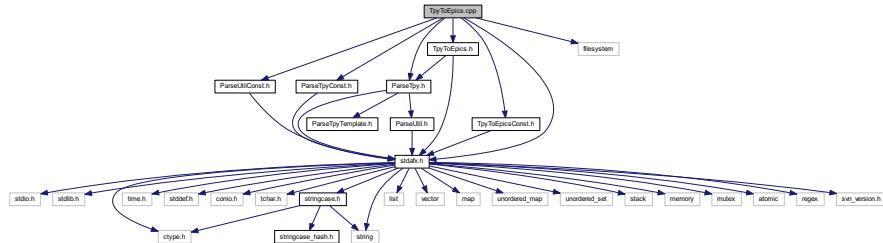
9.27.1 Detailed Description

Header which includes classes to interface with the TCat system and manage TCat symbols.

9.28 TpyToEpics.cpp File Reference

```
#include "stdafx.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "ParseTpy.h"
#include "TpyToEpicsConst.h"
#include "TpyToEpics.h"
#include <filesystem>
```

Include dependency graph for TpyToEpics.cpp:



Namespaces

- [EpicsTpy](#)

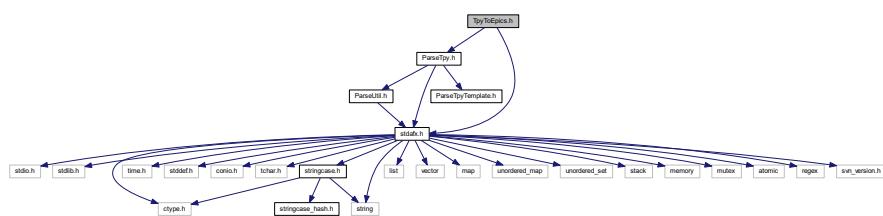
Namespace for tpy-db conversion.

9.28.1 Detailed Description

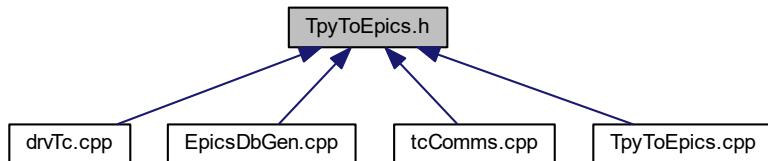
Source for methods that generate EPICs .db file from a .tpy file

9.29 TpyToEpics.h File Reference

```
#include "stdaafx.h"
#include "ParseTpy.h"
Include dependency graph for TpyToEpics.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [EpicsTpy::epics_conversion](#)
Epics conversion.
- class [EpicsTpy::split_io_support](#)
Split IO support.
- class [EpicsTpy::multi_io_support](#)
Multiple IO support.
- class [EpicsTpy::epics_list_processing](#)
List processing.
- struct [EpicsTpy::macro_info](#)

- struct `EpicsTpy::macro_record`

Macro record.
- class `EpicsTpy::epics_macrofiles_processing`

Macro file processing.
- class `EpicsTpy::epics_db_processing`

epics database record processing

Namespaces

- `EpicsTpy`

Namespace for tpy-db conversion.

Typedefs

- using `EpicsTpy::macro_list = std::vector< macro_info >`
- using `EpicsTpy::macro_stack = std::stack< macro_record >`
- using `EpicsTpy::filename_set = std::unordered_set< std::stringcase >`

Enumerations

- enum `EpicsTpy::tc_epics_conv { EpicsTpy::tc_epics_conv::no_conversion, EpicsTpy::tc_epics_conv::no_dot, EpicsTpy::tc_epics_conv::ligo_std, EpicsTpy::tc_epics_conv::ligo_vac }`

Conversion rules for TC/EPICS.
- enum `EpicsTpy::case_type { EpicsTpy::case_type::preserve, EpicsTpy::case_type::upper, EpicsTpy::case_type::lower }`

Case conversion rule enum.
- enum `EpicsTpy::io_filestat { EpicsTpy::io_filestat::closed, read, write }`

enum for file io
- enum `EpicsTpy::listing_type { EpicsTpy::listing_type::standard, EpicsTpy::listing_type::autoburt, EpicsTpy::listing_type::daqini }`

Listing type enum.
- enum `EpicsTpy::macrofile_type { EpicsTpy::macrofile_type::all, EpicsTpy::macrofile_type::fields, EpicsTpy::macrofile_type::error }`
- enum `EpicsTpy::device_support_type { EpicsTpy::device_support_type::opc_name, EpicsTpy::device_support_type::tc_name }`

Device support enum.
- enum `EpicsTpy::string_support_type { EpicsTpy::string_support_type::short_string, EpicsTpy::string_support_type::long_string, vary_string }`

String support enum.
- enum `EpicsTpy::int_support_type { EpicsTpy::int_support_type::int_32, EpicsTpy::int_support_type::int_64, int_auto }`

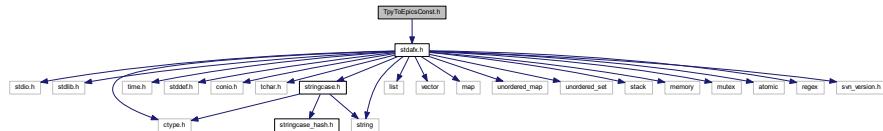
Integer support enum.

9.29.1 Detailed Description

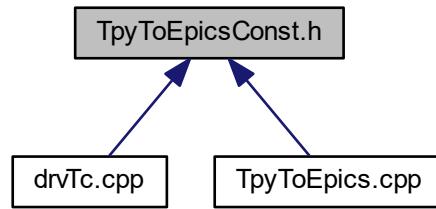
Header which includes classes to convert a parsed TwinCAT tpy into an EPICS database. It includes "ParseTpy.h"

9.30 TpyToEpicsConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for TpyToEpicsConst.h:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpy](#)
Namespace for parsing.
- [EpicsTpy](#)
Namespace for tpy-db conversion.

Variables

- `constexpr int EpicsTpy::MAX_EPICS_CHANNEL = 54`
- `constexpr int EpicsTpy::MAX_EPICS_DESC = 40`
- `constexpr int EpicsTpy::MAX_EPICS_STRING = 40`
- `constexpr int EpicsTpy::MAX_EPICS_LONGSTRING = 65535`
- `constexpr int EpicsTpy::MAX_EPICS_UNIT = 15`
- `constexpr int EpicsTpy::MAX_EPICS_ENUM = 25`
- `const char *const EpicsTpy::EPICS_DB_EGU = "EGU"`
- `const char *const EpicsTpy::EPICS_DB_DESC = "DESC"`
- `const char *const EpicsTpy::EPICS_DB_HOPR = "HOPR"`
- `const char *const EpicsTpy::EPICS_DB_LOPR = "LOPR"`
- `const char *const EpicsTpy::EPICS_DB_DRVH = "DRVH"`
- `const char *const EpicsTpy::EPICS_DB_DRVL = "DRVL"`
- `const char *const EpicsTpy::EPICS_DB_ONAM = "ONAM"`
- `const char *const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"`

- const char *const EpicsTpy::EPICS_DB_PREC = "PREC"
- const char *const EpicsTpy::EPICS_DB_ZRST [16]
- const char *const EpicsTpy::EPICS_DB_ZRVL [16]
- const char *const EpicsTpy::EPICS_DB_SCAN = "SCAN"
- const char *const EpicsTpy::EPICS_DB_INP = "INP"
- const char *const EpicsTpy::EPICS_DB_OUT = "OUT"
- const char *const EpicsTpy::EPICS_DB_TSE = "TSE"
- const char *const EpicsTpy::EPICS_DB_PINI = "PINI"
- const char *const EpicsTpy::EPICS_DB_DTYP = "DTYP"
- const char *const EpicsTpy::EPICS_DB_SIZV = "SIZV"
- const char *const EpicsTpy::EPICS_DB_OSV = "OSV"
- const char *const EpicsTpy::EPICS_DB_ZSV = "ZSV"
- const char *const EpicsTpy::EPICS_DB_COSV = "COSV"
- const char *const EpicsTpy::EPICS_DB_HIHI = "HIHI"
- const char *const EpicsTpy::EPICS_DB_HIGH = "HIGH"
- const char *const EpicsTpy::EPICS_DB_LOW = "LOW"
- const char *const EpicsTpy::EPICS_DB_LOLO = "LOLO"
- const char *const EpicsTpy::EPICS_DB_HYST = "HYST"
- const char *const EpicsTpy::EPICS_DB_HHSV = "HHSV"
- const char *const EpicsTpy::EPICS_DB_HSV = "HSV"
- const char *const EpicsTpy::EPICS_DB_LSV = "LSV"
- const char *const EpicsTpy::EPICS_DB_LLSV = "LLSV"
- const char *const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
- const char *const EpicsTpy::EPICS_DB_MINOR = "MINOR"
- const char *const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
- const char *const EpicsTpy::EPICS_DB_UNSV = "UNSV"
- const char *const EpicsTpy::EPICS_DB_ZRSV [16]
- const char *const EpicsTpy::EPICS_DB_FORBIDDEN []
- const char *const EpicsTpy::EPICS_DB_ALLOWED []
- const char *const EpicsTpy::EPICS_DB_NUMVAL []
- const char *const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"
- constexpr int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4
- constexpr int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2
- constexpr int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
- const char *const EpicsTpy::LIGODAQ_UNIT_NAME = "units"
- const char *const EpicsTpy::LIGODAQ_UNIT_NONE = "none"
- const char *const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
- const char *const EpicsTpy::LIGODAQ_INI_HEADER

9.30.1 Detailed Description

Header which includes EPICS constants needed to generate an EPICS db file from a parsed TwinCAT tpy file.

Index

aaival
 Device support for TwinCAT/ADS, 17

aooval
 Device support for TwinCAT/ADS, 17

access_rights_enum
 plc, 93

add
 plc::BasePLC, 115
 plc::System, 253
 TcComms::tcProcWrite, 270

ADSCallback
 TcComms, 98
 TcComms::TcPLC, 268

airval
 Device support for TwinCAT/ADS, 18

aival
 Device support for TwinCAT/ADS, 17

all
 EPICS utility functions and classes, 60
 Parser utility functions and classes, 33

aorval
 Device support for TwinCAT/ADS, 18

aoval
 Device support for TwinCAT/ADS, 18

arraytype
 TwinCAT tpy file parser, 28

atomic
 Parser utility functions and classes, 33

atomic_string.h, 283

autoburst
 EPICS utility functions and classes, 60

base_record
 ParseTpy::base_record, 110, 111

BasePLCList
 plc, 92

BasePLCPtr
 plc, 92

BaseRecord
 plc::BaseRecord, 123

BaseRecordList
 plc, 92

BaseRecordPtr
 plc, 92

birval
 Device support for TwinCAT/ADS, 18

bival
 Device support for TwinCAT/ADS, 18

borval
 Device support for TwinCAT/ADS, 18

boval
 Device support for TwinCAT/ADS, 18

case_type
 EPICS utility functions and classes, 58

closed
 EPICS utility functions and classes, 60

closePort
 TcComms::TcPLC, 265

compare
 String functions and classes, 50

compareNamesWoNamespace
 ParseTpy, 87

cyclesLeft
 TcComms::TcPLC, 268

daqini
 EPICS utility functions and classes, 60

data_type_enum
 plc, 94

DataValue
 plc::DataValue, 143

dbinfo_list
 InfoPlc::InfoInterface, 207

Device driver functions, 23

Device support for TwinCAT/ADS, 13
 aaival, 17
 aooval, 17
 airval, 18
 aival, 17
 aorval, 18
 aoval, 18
 birval, 18
 bival, 18
 borval, 18
 boval, 18
 epics_record_enum, 17
 epics_record_enumEnd, 18
 eventval, 18
 get_callback_queue_free, 18
 get_callback_queue_highwatermark, 19
 get_callback_queue_overflow, 19
 get_callback_queue_size, 19
 get_callback_queue_used, 20
 histogramval, 18
 invalidval, 18
 isCallback, 22
 isPassive, 22
 linkRecord, 21
 longinval, 18

longoutval, 18
 lsival, 18
 lsoval, 18
 mbbiDirectrval, 18
 mbbiDirectval, 18
 mbbirval, 18
 mbbival, 18
 mbboDirectrval, 18
 mbboDirectval, 18
 mbborval, 18
 mbboval, 18
 register_devsup, 21
 set_callback_queue_highwatermark_reset, 21
 stringintval, 18
 stringoutval, 18
 the_register_devsup, 22
 waveformval, 18
device_support_type
 EPICS utility functions and classes, 59
DevTc, 75
 tc_regex, 77
 tcAlias, 77
 tcInfoPrefix, 77
 tcList, 79
 tcLoadRecords, 79
 tcMacro, 79
 tcPrintVal, 80
 tcPrintVals, 80
 tcSetScanRate, 80
devTc.cpp, 284
devTc.h, 285
DevTc::devTcDefIn< RecType >, 164
DevTc::devTcDeflo< RecType >, 166
DevTc::devTcDefOut< RecType >, 168
DevTc::devTcDefWaveformIn< RecType >, 169
DevTc::epics_record_traits< RecType >, 190
DevTc::epics_tc_db_processing, 191
 epics_tc_db_processing, 192
 getopt, 193
 operator(), 193
 process_list, 193
 process_lists, 194
 process_macro, 194
 process_macros, 195
DevTc::EpicsInterface, 195
DevTc::register_devsup, 237
DevTc::tcRegisterTolocShell, 271
dimension
 TwinCAT tpy file parser, 26
dimensions
 TwinCAT tpy file parser, 27
done
 Info interface classes and functions, 25
drvTc.cpp, 287
drvTc.h, 289
dtBinary
 plc, 94
dtBool
 plc, 94
dtDouble
 plc, 94
dtFloat
 plc, 94
dtInt16
 plc, 94
dtInt32
 plc, 94
dtInt64
 plc, 94
dtInt8
 plc, 94
dtInvalid
 plc, 94
dtString
 plc, 94
dtUInt16
 plc, 94
dtUInt32
 plc, 94
dtUInt64
 plc, 94
dtUInt8
 plc, 94
dtWString
 plc, 94
enum_map
 TwinCAT tpy file parser, 27
enum_pair
 TwinCAT tpy file parser, 27
enumtype
 TwinCAT tpy file parser, 28
EPICS constants, 62
 EPICS_DB_ALLOWED, 63
 EPICS_DB_COSV, 63
 EPICS_DB_DESC, 63
 EPICS_DB_DRVH, 63
 EPICS_DB_DRVL, 63
 EPICS_DB_DTYP, 64
 EPICS_DB_EGU, 64
 EPICS_DB_FORBIDDEN, 64
 EPICS_DB_HHSV, 64
 EPICS_DB_HIGH, 64
 EPICS_DB_HIHI, 65
 EPICS_DB_HOPR, 65
 EPICS_DB_HSV, 65
 EPICS_DB_HYST, 65
 EPICS_DB_INP, 65
 EPICS_DB_LLSV, 66
 EPICS_DB_LOLO, 66
 EPICS_DB_LOPR, 66
 EPICS_DB_LOW, 66
 EPICS_DB_LSV, 66
 EPICS_DB_MAJOR, 67
 EPICS_DB_MINOR, 67
 EPICS_DB_NOALARM, 67
 EPICS_DB_NUMVAL, 67

EPICS_DB_ONAM, 68
EPICS_DB_OSV, 68
EPICS_DB_OUT, 68
EPICS_DB_PINI, 68
EPICS_DB_PREC, 68
EPICS_DB_SCAN, 69
EPICS_DB_SIZV, 69
EPICS_DB_TSE, 69
EPICS_DB_UNSV, 69
EPICS_DB_ZNAM, 69
EPICS_DB_ZRST, 70
EPICS_DB_ZRSV, 70
EPICS_DB_ZRVL, 70
EPICS_DB_ZSV, 70
LIGODAQ_DATATYPE_DEFAULT, 71
LIGODAQ_DATATYPE_FLOAT, 71
LIGODAQ_DATATYPE_INT32, 71
LIGODAQ_DATATYPE_NAME, 71
LIGODAQ_INI_HEADER, 72
LIGODAQ_UNIT_DEFAULT, 72
LIGODAQ_UNIT_NAME, 72
LIGODAQ_UNIT_NONE, 72
MAX_EPICS_CHANNEL, 73
MAX_EPICS_DESC, 73
MAX_EPICS_ENUM, 73
MAX_EPICS_LONGSTRING, 73
MAX_EPICS_STRING, 73
MAX_EPICS_UNIT, 74
EPICS utility functions and classes, 57
all, 60
autoburt, 60
case_type, 58
closed, 60
daqini, 60
device_support_type, 59
errors, 60
fields, 60
filename_set, 58
int_32, 59
int_64, 59
int_support_type, 59
io_filestat, 59
ligo_std, 61
ligo_vac, 61
listing_type, 60
long_string, 61
lower, 59
macro_list, 58
macro_stack, 58
macrofile_type, 60
no_conversion, 61
no_dot, 61
opc_name, 59
preserve, 59
short_string, 61
standard, 60
string_support_type, 60
tc_epics_conv, 61
tc_name, 59
upper, 59
epics_conversion
 EpicsTpy::epics_conversion, 172, 173
EPICS_DB_ALLOWED
 EPICS constants, 63
EPICS_DB_COSV
 EPICS constants, 63
EPICS_DB_DESC
 EPICS constants, 63
EPICS_DB_DRVH
 EPICS constants, 63
EPICS_DB_DRVL
 EPICS constants, 63
EPICS_DB_DTYP
 EPICS constants, 64
EPICS_DB_EGU
 EPICS constants, 64
EPICS_DB_FORBIDDEN
 EPICS constants, 64
EPICS_DB_HHSV
 EPICS constants, 64
EPICS_DB_HIGH
 EPICS constants, 64
EPICS_DB_HIHI
 EPICS constants, 65
EPICS_DB_HOPR
 EPICS constants, 65
EPICS_DB_HSV
 EPICS constants, 65
EPICS_DB_HYST
 EPICS constants, 65
EPICS_DB_INP
 EPICS constants, 65
EPICS_DB_LLSV
 EPICS constants, 66
EPICS_DB_LOLO
 EPICS constants, 66
EPICS_DB_LOPR
 EPICS constants, 66
EPICS_DB_LOW
 EPICS constants, 66
EPICS_DB_LSV
 EPICS constants, 66
EPICS_DB_MAJOR
 EPICS constants, 67
EPICS_DB_MINOR
 EPICS constants, 67
EPICS_DB_NOALARM
 EPICS constants, 67
EPICS_DB_NUMVAL
 EPICS constants, 67
EPICS_DB_ONAM
 EPICS constants, 68
EPICS_DB_OSV
 EPICS constants, 68
EPICS_DB_OUT
 EPICS constants, 68

EPICS_DB_PINI
 EPICS constants, 68
 EPICS_DB_PREC
 EPICS constants, 68
 epics_db_processing
 EpicsTpy::epics_db_processing, 176
 EPICS_DB_SCAN
 EPICS constants, 69
 EPICS_DB_SIZV
 EPICS constants, 69
 EPICS_DB_TSE
 EPICS constants, 69
 EPICS_DB_UNSV
 EPICS constants, 69
 EPICS_DB_ZNAM
 EPICS constants, 69
 EPICS_DB_ZRST
 EPICS constants, 70
 EPICS_DB_ZRSV
 EPICS constants, 70
 EPICS_DB_ZRVL
 EPICS constants, 70
 EPICS_DB_ZSV
 EPICS constants, 70
 epics_list_processing
 EpicsTpy::epics_list_processing, 182
 epics_macrofiles_processing
 EpicsTpy::epics_macrofiles_processing, 187
 epics_record_enum
 Device support for TwinCAT/ADS, 17
 epics_record_enumEnd
 Device support for TwinCAT/ADS, 18
 epics_tc_db_processing
 DevTc::epics_tc_db_processing, 192
 EpicsDbGen.cpp, 290
 main, 290
 EpicsTpy, 81
 EpicsTpy::epics_conversion, 171
 epics_conversion, 172, 173
 getopt, 173
 to_epics, 174
 EpicsTpy::epics_db_processing, 175
 epics_db_processing, 176
 getopt, 177
 my getopt, 177
 operator(), 178
 process_field_alarm, 179
 process_field_numeric, 179, 180
 process_field_string, 180
 EpicsTpy::epics_list_processing, 181
 epics_list_processing, 182
 getopt, 183
 my getopt, 183
 operator(), 184
 EpicsTpy::epics_macrofiles_processing, 185
 epics_macrofiles_processing, 187
 getopt, 187
 my getopt, 188
 operator(), 189
 to_filename, 189
 EpicsTpy::macro_info, 211
 EpicsTpy::macro_record, 212
 EpicsTpy::multi_io_support, 216
 getopt, 219
 multi_io_support, 218
 EpicsTpy::split_io_support, 241
 getopt, 244
 increment, 245
 operator=, 245
 split_io_support, 243, 244
 eq
 String functions and classes, 51
 erase
 plc::BasePLC, 116
 errorPrintf
 TcComms, 98
 errors
 EPICS utility functions and classes, 60
 eventval
 Device support for TwinCAT/ADS, 18
 fields
 EPICS utility functions and classes, 60
 filename_set
 EPICS utility functions and classes, 58
 find
 plc::BasePLC, 116
 for_each
 plc::BasePLC, 116, 117
 plc::System, 254
 forever
 Info interface classes and functions, 25
 functionblock
 TwinCAT tpy file parser, 28
 get
 ParseTpy::ads_routing_info, 102
 ParseUtil::memory_location, 215
 ParseUtil::process_arg_tc, 235
 get_callback_queue_free
 Device support for TwinCAT/ADS, 18
 get_callback_queue_highwatermark
 Device support for TwinCAT/ADS, 19
 get_callback_queue_overflow
 Device support for TwinCAT/ADS, 19
 get_callback_queue_size
 Device support for TwinCAT/ADS, 19
 get_callback_queue_used
 Device support for TwinCAT/ADS, 20
 get_decoration
 ParseTpy, 87
 get_full
 InfoPlc::process_arg_info, 232
 ParseUtil::process_arg, 230
 ParseUtil::process_arg_tc, 235
 get_infodb
 InfoPlc::InfoInterface, 205

get_next
 plc::BasePLC, 117

get_pointer
 ParseTpy, 87

get_responseBuffer
 TcComms::TcPLC, 266

get_symbol_name
 InfoPlc::InfoInterface, 206

get_timestamp_unix
 plc::BasePLC, 118

getopt
 DevTc::epics_tc_db_processing, 193
 EpicsTpy::epics_conversion, 173
 EpicsTpy::epics_db_processing, 177
 EpicsTpy::epics_list_processing, 183
 EpicsTpy::epics_macrofiles_processing, 187
 EpicsTpy::multi_io_support, 219
 EpicsTpy::split_io_support, 244
 ParseUtil::tag_processing, 257

GetValid
 plc::DataValue, 143

histogramval
 Device support for TwinCAT/ADS, 18

ignore
 Parser utility functions and classes, 33

increment
 EpicsTpy::split_io_support, 245

Info interface classes and functions, 24
 done, 25
 forever, 25
 info_dbrecord_type, 24
 once, 25
 update_enum, 24

info_dbrecord_type
 Info interface classes and functions, 24

InfoInterface
 InfoPlc::InfoInterface, 205

InfoPlc, 83

infoPlc.cpp, 291

infoPlc.h, 291

InfoPlc::InfoInterface, 200
 dbinfo_list, 207
 get_infodb, 205
 get_symbol_name, 206

InfoInterface, 205
 printVal, 206

InfoPlc::process_arg_info, 231
 get_full, 232
 process_arg_info, 232

infoPlcTemplate.h, 292

Init
 plc::DataValue, 144

int_32
 EPICS utility functions and classes, 59

int_64
 EPICS utility functions and classes, 59

int_support_type

 EPICS utility functions and classes, 59

Interface
 plc::Interface, 209

InterfacePtr
 plc, 93

invalidval
 Device support for TwinCAT/ADS, 18

io_filestat
 EPICS utility functions and classes, 59

iocMain.cpp, 293
 main, 293

isCallback
 Device support for TwinCAT/ADS, 22

isPassive
 Device support for TwinCAT/ADS, 22

item_list
 TwinCAT tpy file parser, 27

ligo_std
 EPICS utility functions and classes, 61

ligo_vac
 EPICS utility functions and classes, 61

LIGODAQ_DATATYPE_DEFAULT
 EPICS constants, 71

LIGODAQ_DATATYPE_FLOAT
 EPICS constants, 71

LIGODAQ_DATATYPE_INT32
 EPICS constants, 71

LIGODAQ_DATATYPE_NAME
 EPICS constants, 71

LIGODAQ_INI_HEADER
 EPICS constants, 72

LIGODAQ_UNIT_DEFAULT
 EPICS constants, 72

LIGODAQ_UNIT_NAME
 EPICS constants, 72

LIGODAQ_UNIT_NONE
 EPICS constants, 72

linkRecord
 Device support for TwinCAT/ADS, 21

listing_type
 EPICS utility functions and classes, 60

long_string
 EPICS utility functions and classes, 61

longinval
 Device support for TwinCAT/ADS, 18

longoutval
 Device support for TwinCAT/ADS, 18

lower
 EPICS utility functions and classes, 59

lival
 Device support for TwinCAT/ADS, 18

lsoval
 Device support for TwinCAT/ADS, 18

lt
 String functions and classes, 51, 52

macro_list
 EPICS utility functions and classes, 58

macro_stack
 EPICS utility functions and classes, 58

macrofile_type
 EPICS utility functions and classes, 60

main
 EpicsDbGen.cpp, 290
 iocMain.cpp, 293
 ParseTpyInfo.cpp, 299

MAX_EPICS_CHANNEL
 EPICS constants, 73

MAX_EPICS_DESC
 EPICS constants, 73

MAX_EPICS_ENUM
 EPICS constants, 73

MAX_EPICS_LONGSTRING
 EPICS constants, 73

MAX_EPICS_STRING
 EPICS constants, 73

MAX_EPICS_UNIT
 EPICS constants, 74

mbbiDirectrv1
 Device support for TwinCAT/ADS, 18

mbbiDirectrv1
 Device support for TwinCAT/ADS, 18

mbbirval
 Device support for TwinCAT/ADS, 18

mbbival
 Device support for TwinCAT/ADS, 18

mbboDirectrv1
 Device support for TwinCAT/ADS, 18

mbboDirectrv1
 Device support for TwinCAT/ADS, 18

mbborval
 Device support for TwinCAT/ADS, 18

mbboval
 Device support for TwinCAT/ADS, 18

memory_location
 ParseUtil::memory_location, 214, 215

multi_io_support
 EpicsTpy::multi_io_support, 218

mygetopt
 EpicsTpy::epics_db_processing, 177
 EpicsTpy::epics_list_processing, 183
 EpicsTpy::epics_macrofiles_processing, 188

mysize
 plc::DataValue, 161

name_parse
 ParseTpy::parserinfo_type, 226

ne
 String functions and classes, 52

no_change
 Parser utility functions and classes, 33

no_conversion
 EPICS utility functions and classes, 61

no_dot
 EPICS utility functions and classes, 61

once
 Info interface classes and functions, 25

OPC property constants, 35
 OPC_NAME_ALIAS, 36
 OPC_PROP_ALIAS, 36
 OPC_PROP_ALMAREAS, 36
 OPC_PROP_ALMCONDITION, 36
 OPC_PROP_ALMCOSV, 37
 OPC_PROP_ALMDB, 37
 OPC_PROP_ALMDEV, 37
 OPC_PROP_ALMFFSV, 37
 OPC_PROP_ALMH, 37
 OPC_PROP_ALMHELP, 38
 OPC_PROP_ALMH, 38
 OPC_PROP_ALMHHSV, 38
 OPC_PROP_ALMHHSV, 38
 OPC_PROP_ALML, 38
 OPC_PROP_ALMLIMIT, 39
 OPC_PROP_ALMLL, 39
 OPC_PROP_ALMLLSV, 39
 OPC_PROP_ALMLSV, 39
 OPC_PROP_ALMOSV, 39
 OPC_PROP_ALMPRIMARYAREA, 40
 OPC_PROP_ALMROC, 40
 OPC_PROP_ALMSTAT, 40
 OPC_PROP_ALMUNSV, 40
 OPC_PROP_ALMZRSV, 40
 OPC_PROP_ALMZSV, 41
 OPC_PROP_AVI, 41
 OPC_PROP_BGC, 41
 OPC_PROP_BLINK, 41
 OPC_PROP_BMP, 41
 OPC_PROP_CDT, 42
 OPC_PROP_CLOSE, 42
 OPC_PROP_DESC, 42
 OPC_PROP_DTYP, 42
 OPC_PROP_FFST, 42
 OPC_PROP_FGC, 43
 OPC_PROP_FIELD_BEG, 43
 OPC_PROP_FIELD_END, 43
 OPC_PROP_HIEU, 43
 OPC_PROP_HIRANGE, 43
 OPC_PROP_HTML, 44
 OPC_PROP_INOUT, 44
 OPC_PROP_INPUT, 44
 OPC_PROP_LOEU, 44
 OPC_PROP_LORANGE, 44
 OPC_PROP_OPEN, 45
 OPC_PROP_OUTPUT, 45
 OPC_PROP_PINI, 45
 OPC_PROP_PLNAME, 45
 OPC_PROP_PREC, 45
 OPC_PROP_QUALITY, 46
 OPC_PROP_RECTYPE, 46
 OPC_PROP_RIGHTS, 46
 OPC_PROP_SCANRATE, 46
 OPC_PROP_SERVER, 46
 OPC_PROP SND, 47
 OPC_PROP_TIME, 47

OPC_PROP_TIMEZONE, 47
OPC_PROP_TSE, 47
OPC_PROP_UNIT, 47
OPC_PROP_VALUE, 48
OPC_PROP_ZRST, 48
opc_enum
 Parser utility functions and classes, 32
opc_name
 EPICS utility functions and classes, 59
OPC_NAME_ALIAS
 OPC property constants, 36
OPC_PROP_ALIAS
 OPC property constants, 36
OPC_PROP_ALMAREAS
 OPC property constants, 36
OPC_PROP_ALMCONDITION
 OPC property constants, 36
OPC_PROP_ALMCOSV
 OPC property constants, 37
OPC_PROP_ALMDB
 OPC property constants, 37
OPC_PROP_ALMDEV
 OPC property constants, 37
OPC_PROP_ALMFFSV
 OPC property constants, 37
OPC_PROP_ALMH
 OPC property constants, 37
OPC_PROP_ALMHELP
 OPC property constants, 38
OPC_PROP_ALMHHSV
 OPC property constants, 38
OPC_PROP_ALMHHSV
 OPC property constants, 38
OPC_PROP_ALML
 OPC property constants, 38
OPC_PROP_ALMLIMIT
 OPC property constants, 39
OPC_PROP_ALMLL
 OPC property constants, 39
OPC_PROP_ALMLLSV
 OPC property constants, 39
OPC_PROP_ALMLSV
 OPC property constants, 39
OPC_PROP_ALMOSV
 OPC property constants, 39
OPC_PROP_ALMPRIMARYAREA
 OPC property constants, 40
OPC_PROP_ALMROC
 OPC property constants, 40
OPC_PROP_ALMSTAT
 OPC property constants, 40
OPC_PROP_ALMUNSV
 OPC property constants, 40
OPC_PROP_ALMZRSV
 OPC property constants, 40
OPC_PROP_ALMZSV
 OPC property constants, 40
OPC_PROP_PROPERTY
 OPC property constants, 41
OPC_PROP_AVI
 OPC property constants, 41
OPC_PROP_BGC
 OPC property constants, 41
OPC_PROP_BLINK
 OPC property constants, 41
OPC_PROP_BMP
 OPC property constants, 41
OPC_PROP_CDT
 OPC property constants, 42
OPC_PROP_CLOSE
 OPC property constants, 42
OPC_PROP_DESC
 OPC property constants, 42
OPC_PROP_DTYP
 OPC property constants, 42
OPC_PROP_FFST
 OPC property constants, 42
OPC_PROP_FGC
 OPC property constants, 43
OPC_PROP_FIELD_BEG
 OPC property constants, 43
OPC_PROP_FIELD_END
 OPC property constants, 43
OPC_PROP_HIEU
 OPC property constants, 43
OPC_PROP_HIRANGE
 OPC property constants, 43
OPC_PROP_HTML
 OPC property constants, 44
OPC_PROP_INOUT
 OPC property constants, 44
OPC_PROP_INPUT
 OPC property constants, 44
OPC_PROP_LOEU
 OPC property constants, 44
OPC_PROP_LORANGE
 OPC property constants, 44
OPC_PROP_OPEN
 OPC property constants, 45
OPC_PROP_OUTPUT
 OPC property constants, 45
OPC_PROP_PINI
 OPC property constants, 45
OPC_PROP_PLNAME
 OPC property constants, 45
OPC_PROP_PREC
 OPC property constants, 45
OPC_PROP_QUALITY
 OPC property constants, 46
OPC_PROP_RECTYPE
 OPC property constants, 46
OPC_PROP_RIGHTS
 OPC property constants, 46
OPC_PROP_SCANRATE
 OPC property constants, 46
OPC_PROP_SERVER

OPC property constants, 46
OPC_PROP_SND
 OPC property constants, 47
OPC_PROP_TIME
 OPC property constants, 47
OPC_PROP_TIMEZONE
 OPC property constants, 47
OPC_PROP_TSE
 OPC property constants, 47
OPC_PROP_UNIT
 OPC property constants, 47
OPC_PROP_VALUE
 OPC property constants, 48
OPC_PROP_ZRST
 OPC property constants, 48
operator()
 DevTc::epics_tc_db_processing, 193
 EpicsTpy::epics_db_processing, 178
 EpicsTpy::epics_list_processing, 184
 EpicsTpy::epics_macrofiles_processing, 189
 std::std::hash< std::stringcase >, 198
 std::std::hash< std::wstringcase >, 199
operator=
 EpicsTpy::split_io_support, 245
 std::atomic_string< stringT >, 108
optarg
 ParseUtil::optarg, 222
optimizeRequests
 TcComms::TcPLC, 266

parse
 ParseUtil::optarg, 223
parse_finish
 ParseTpy::tpy_file, 273
parse_rules
 ParseUtil::replacement_rules, 240
Parser utility functions and classes, 31
 all, 33
 atomic, 33
 ignore, 33
 no_change, 33
 opc_enum, 32
 process_substitution_enum, 33
 process_tag_enum, 33
 process_type_enum, 33
 property_el, 32
 property_map, 32
 pt_binary, 34
 pt_bool, 34
 pt_enum, 34
 pt_int, 34
 pt_invalid, 34
 pt_real, 34
 pt_string, 34
 publish, 33
 replacement_table, 32
 silent, 33
 standard, 33
 structured, 33

 substitution_map, 32
ParseTpy, 84
 compareNamesWoNamespace, 87
 get_decoration, 87
 get_pointer, 87
ParseTpy.cpp, 294
ParseTpy.h, 295
ParseTpy::ads_routing_info, 101
 get, 102
 set, 103
ParseTpy::base_record, 109
 base_record, 110, 111
ParseTpy::compiler_info, 137
ParseTpy::item_record, 210
ParseTpy::parserinfo_type, 224
 name_parse, 226
ParseTpy::project_record, 236
ParseTpy::symbol_record, 250
ParseTpy::tpy_file, 272
 parse_finish, 273
 process_array, 274
 process_symbols, 275
 process_type_tree, 276, 277
ParseTpy::type_map, 278
ParseTpy::type_record, 279
ParseTpyConst.h, 296
ParseTpyInfo.cpp, 299
 main, 299
ParseTpyTemplate.h, 300
ParseUtil, 88
ParseUtil.cpp, 300
ParseUtil.h, 301
ParseUtil::bit_location, 134
ParseUtil::memory_location, 213
 get, 215
 memory_location, 214, 215
 set, 215
 set_section, 216
ParseUtil::opc_list, 220
ParseUtil::optarg, 221
 optarg, 222
 parse, 223
ParseUtil::parserinfo_type, 226
 tag, 227
ParseUtil::process_arg, 228
 get_full, 230
 process_arg, 230
ParseUtil::process_arg_tc, 233
 get, 235
 get_full, 235
 process_arg_tc, 234
ParseUtil::replacement_rules, 238
 parse_rules, 240
 replacement_rules, 240
ParseUtil::substitution, 246
 substitution, 247
ParseUtil::substitution_list, 248
 query_substitution, 249

ParseUtil::tag_processing, 255
getopt, 257
tag_processing, 257
ParseUtil::variable_name, 281
ParseUtilConst.h, 302
plc, 90
 access_rights_enum, 93
 BasePLCList, 92
 BasePLCPtr, 92
 BaseRecordList, 92
 BaseRecordPtr, 92
 data_type_enum, 94
 dtBinary, 94
 dtBool, 94
 dtDouble, 94
 dtFloat, 94
 dtInt16, 94
 dtInt32, 94
 dtInt64, 94
 dtInt8, 94
 dtInvalid, 94
 dtString, 94
 dtUInt16, 94
 dtUInt32, 94
 dtUInt64, 94
 dtUInt8, 94
 dtWString, 94
 InterfacePtr, 93
 read_only, 93
 read_write, 93
 reset_and_read, 94, 95
 scanner_thread_args, 93
 ScannerProc, 95
 scannerThread, 95
 write_and_test, 96
 write_only, 93
plc::BasePLC, 112
 add, 115
 erase, 116
 find, 116
 for_each, 116, 117
 get_next, 117
 get_timestamp_unix, 118
 plc_data_set_valid, 118
 printRecord, 118
 records, 120
 reserve, 119
 start, 119
 user_data_set_valid, 119
plc::BaseRecord, 120
 BaseRecord, 123
 PlcGetValid, 124
 PlcPush, 124
 PlcRead, 125
 PlcReadBinary, 126
 PlcSetValid, 126
 PlcWrite, 127, 128
 PlcWriteBinary, 128
 UserGetValid, 129
 UserPush, 129
 UserRead, 129, 130
 UserReadBinary, 131
 UserSetValid, 131
 UserWrite, 131, 132
 UserWriteBinary, 133
plc::DataValue, 140
 DataValue, 143
 GetValid, 143
 Init, 144
 mysize, 161
 PlcGetValid, 144
 PlcRead, 144, 145
 PlcReadBinary, 146
 PlcSetValid, 146
 PlcWrite, 147, 148
 PlcWriteBinary, 148
 Read, 148–150
 ReadBinary, 150
 SetValid, 151
 UserGetValid, 151
 UserRead, 152, 153
 UserReadBinary, 153
 UserSetValid, 154
 UserWrite, 154, 156
 UserWriteBinary, 156
 Write, 157, 159, 160
 WriteBinary, 160
plc::DataValueTraits< T >, 161
plc::DataValueTypeDef, 162
plc::Interface, 208
 Interface, 209
 printVal, 210
plc::System, 252
 add, 253
 for_each, 254
 printVal, 255
plc_data_set_valid
 plc::BasePLC, 118
plcBase.cpp, 304
plcBase.h, 305
plcBaseTemplate.h, 307
PlcGetValid
 plc::BaseRecord, 124
 plc::DataValue, 144
PlcPush
 plc::BaseRecord, 124
PlcRead
 plc::BaseRecord, 125
 plc::DataValue, 144, 145
PlcReadBinary
 plc::BaseRecord, 126
 plc::DataValue, 146
PlcSetValid
 plc::BaseRecord, 126
 plc::DataValue, 146
PlcWrite

plc::BaseRecord, 127, 128
 plc::DataValue, 147, 148
PlcWriteBinary
 plc::BaseRecord, 128
 plc::DataValue, 148
preserve
 EPICS utility functions and classes, 59
printRecord
 plc::BasePLC, 118
 TcComms::TcPLC, 266
printVal
 InfoPlc::InfoInterface, 206
 plc::Interface, 210
 plc::System, 255
 TcComms::TCatInterface, 262
process_arg
 ParseUtil::process_arg, 230
process_arg_info
 InfoPlc::process_arg_info, 232
process_arg_tc
 ParseUtil::process_arg_tc, 234
process_array
 ParseTpy::tpy_file, 274
process_field_alarm
 EpicsTpy::epics_db_processing, 179
process_field_numeric
 EpicsTpy::epics_db_processing, 179, 180
process_field_string
 EpicsTpy::epics_db_processing, 180
process_list
 DevTc::epics_tc_db_processing, 193
process_lists
 DevTc::epics_tc_db_processing, 194
process_macro
 DevTc::epics_tc_db_processing, 194
process_macros
 DevTc::epics_tc_db_processing, 195
process_substitution_enum
 Parser utility functions and classes, 33
process_symbols
 ParseTpy::tpy_file, 275
process_tag_enum
 Parser utility functions and classes, 33
process_type_enum
 Parser utility functions and classes, 33
process_type_tree
 ParseTpy::tpy_file, 276, 277
property_el
 Parser utility functions and classes, 32
property_map
 Parser utility functions and classes, 32
pt_binary
 Parser utility functions and classes, 34
pt_bool
 Parser utility functions and classes, 34
pt_enum
 Parser utility functions and classes, 34
pt_int
 Parser utility functions and classes, 34
Parser utility functions and classes, 34
pt_invalid
 Parser utility functions and classes, 34
pt_real
 Parser utility functions and classes, 34
pt_string
 Parser utility functions and classes, 34
publish
 Parser utility functions and classes, 33
query_substitution
 ParseUtil::substitution_list, 249
Read
 plc::DataValue, 148–150
read_only
 plc, 93
read_ptr
 TcComms::tcProcWrite, 270
read_write
 plc, 93
ReadBinary
 plc::DataValue, 150
records
 plc::BasePLC, 120
register_devsup
 Device support for TwinCAT/ADS, 21
replacement_rules
 ParseUtil::replacement_rules, 240
replacement_table
 Parser utility functions and classes, 32
req
 TcComms::tcProcWrite, 271
reserve
 plc::BasePLC, 119
reset_and_read
 plc, 94, 95
RouterCall
 TcComms, 99
 TcComms::AmsRouterNotification, 104
scanner_thread_args
 plc, 93
ScannerProc
 plc, 95
scannerThread
 plc, 95
set
 ParseTpy::ads_routing_info, 103
 ParseUtil::memory_location, 215
set_addr
 TcComms::TcPLC, 267
set_ads_state
 TcComms::TcPLC, 267
set_callback_queue_highwatermark_reset
 Device support for TwinCAT/ADS, 21
set_section
 ParseUtil::memory_location, 216
SetValid

plc::DataValue, 151
short_string
 EPICS utility functions and classes, 61
silent
 Parser utility functions and classes, 33
simple
 TwinCAT tpy file parser, 28
split_io_support
 EpicsTpyp::split_io_support, 243, 244
split_string
 String functions and classes, 53
standard
 EPICS utility functions and classes, 60
 Parser utility functions and classes, 33
start
 plc::BasePLC, 119
std::atomic< string >, 105
std::atomic< wstring >, 106
std::atomic_string< stringT >, 107
 operator=, 108
std::case_char_traits, 135
std::case_wchar_traits, 136
std::std::hash< std::stringcase >, 198
 operator(), 198
std::std::hash< std::wstringcase >, 199
 operator(), 199
stdafx.cpp, 307
stdafx.h, 308
String functions and classes, 49
 compare, 50
 eq, 51
 lt, 51, 52
 ne, 52
 split_string, 53
 stringcase, 49
 strncasecmp, 53
 trim_space, 54
 wcsncasewcmp, 55
 wstringcase, 49
string_support_type
 EPICS utility functions and classes, 60
stringcase
 String functions and classes, 49
stringcase.h, 308
stringcase_hash.h, 310
stringintval
 Device support for TwinCAT/ADS, 18
stringoutval
 Device support for TwinCAT/ADS, 18
strncasecmp
 String functions and classes, 53
structtype
 TwinCAT tpy file parser, 28
structured
 Parser utility functions and classes, 33
substitution
 ParseUtil::substitution, 247
substitution_map
 Parser utility functions and classes, 32
symbol_list
 TwinCAT tpy file parser, 27
syminfo_processing, 251
tag
 ParseUtil::parserinfo_type, 227
tag_processing
 ParseUtil::tag_processing, 257
tc_epics_conv
 EPICS utility functions and classes, 61
tc_name
 EPICS utility functions and classes, 59
tc_regex
 DevTc, 77
tcAlias
 DevTc, 77
TCatInterface
 TcComms::TCatInterface, 261
TcComms, 97
 ADSCallback, 98
 errorPrintf, 98
 RouterCall, 99
tcComms.cpp, 310
tcComms.h, 311
TcComms::AmsRouterNotification, 104
 RouterCall, 104
TcComms::DataPar, 139
TcComms::TCatInterface, 258
 printVal, 262
 TCatInterface, 261
TcComms::TcPLC, 262
 ADSCallback, 268
 closePort, 265
 cyclesLeft, 268
 get_responseBuffer, 266
 optimizeRequests, 266
 printRecord, 266
 set_addr, 267
 set_ads_state, 267
 update_scanner, 267
TcComms::tcProcWrite, 268
 add, 270
 read_ptr, 270
 req, 271
tcInfoPrefix
 DevTc, 77
tcList
 DevTc, 79
tcLoadRecords
 DevTc, 79
tcMacro
 DevTc, 79
tcPrintVal
 DevTc, 80
tcPrintVals
 DevTc, 80
tcSetScanRate
 DevTc, 80

the_register_devsup
 Device support for TwinCAT/ADS, 22

to_epics
 EpicsTpy::epics_conversion, 174

to_filename
 EpicsTpy::epics_macrofiles_processing, 189

TpyToEpics.cpp, 312

TpyToEpics.h, 313

TpyToEpicsConst.h, 315

trim_space
 String functions and classes, 54

TwinCAT read/write scanning, 56

TwinCAT tpy file parser, 26

- arraytype, 28
- dimension, 26
- dimensions, 27
- enum_map, 27
- enum_pair, 27
- enumtype, 28
- functionblock, 28
- item_list, 27
- simple, 28
- structtype, 28
- symbol_list, 27
- type_enum, 28
- type_multipmap, 28
- unknown, 28

type_enum
 TwinCAT tpy file parser, 28

type_multipmap
 TwinCAT tpy file parser, 28

unknown
 TwinCAT tpy file parser, 28

update_enum
 Info interface classes and functions, 24

update_scanner
 TcComms::TcPLC, 267

upper
 EPICS utility functions and classes, 59

user_data_set_valid
 plc::BasePLC, 119

UserGetValid
 plc::BaseRecord, 129
 plc::DataValue, 151

UserPush
 plc::BaseRecord, 129

UserRead
 plc::BaseRecord, 129, 130
 plc::DataValue, 152, 153

UserReadBinary
 plc::BaseRecord, 131
 plc::DataValue, 153

UserSetValid
 plc::BaseRecord, 131
 plc::DataValue, 154

UserWrite
 plc::BaseRecord, 131, 132
 plc::DataValue, 154, 156

UserWriteBinary
 plc::BaseRecord, 133
 plc::DataValue, 156

waveformval
 Device support for TwinCAT/ADS, 18

wcsncasecmp
 String functions and classes, 55

Write
 plc::DataValue, 157, 159, 160

write_and_test
 plc, 96

write_only
 plc, 93

WriteBinary
 plc::DataValue, 160

wstringcase
 String functions and classes, 49

XML tpy file constants, 29